Graph Transformer

Liang Wang

Nov. 3, 2022

- Background
  - Transformer
  - Massage Passing GNNs (MPNNs)
- Graph Transformer
  - Localized Graph Transformer (Transformer-based MPNN)
    self-attention on the neighbors of each node
  - Global Graph Transformer
    self-attention on the entire graph
    - Structural/Positional Encoding
    - Scaling to Large Graphs
- Quick Look at ICLR 2023 Submissions

Background

Transformer, NeurIPS, 2017

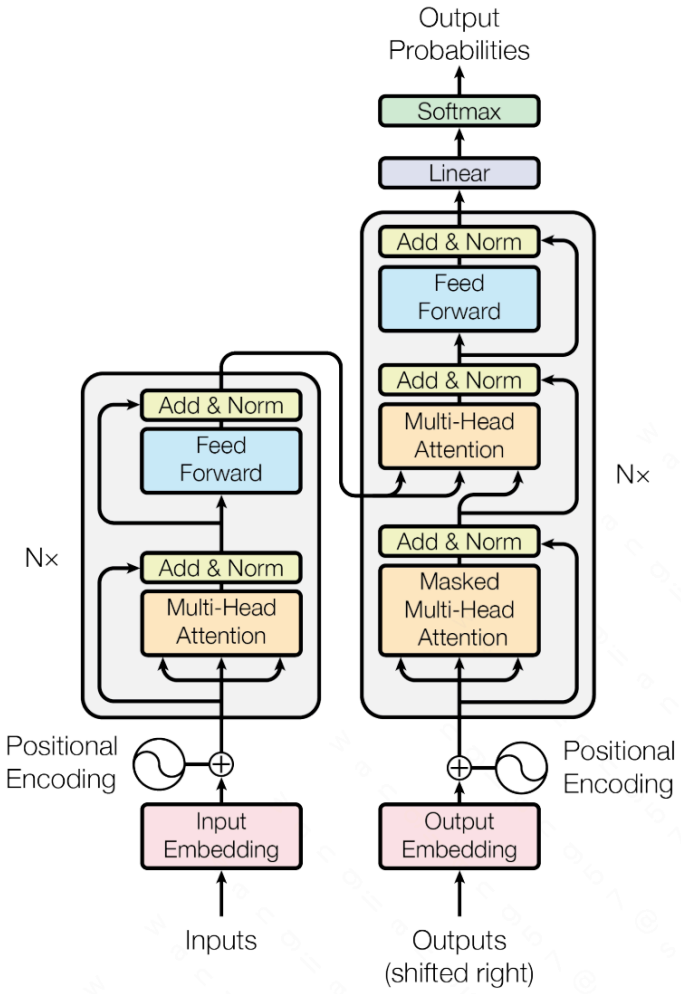
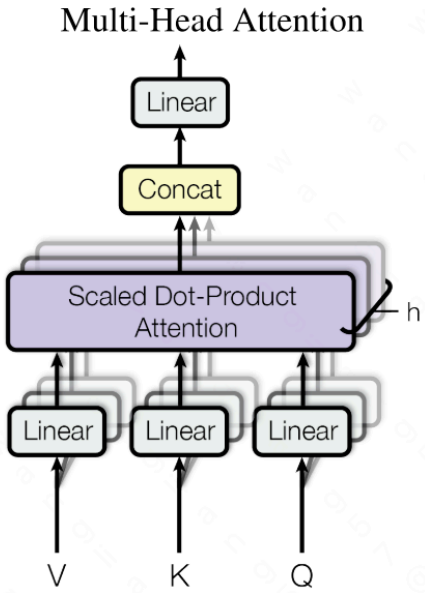
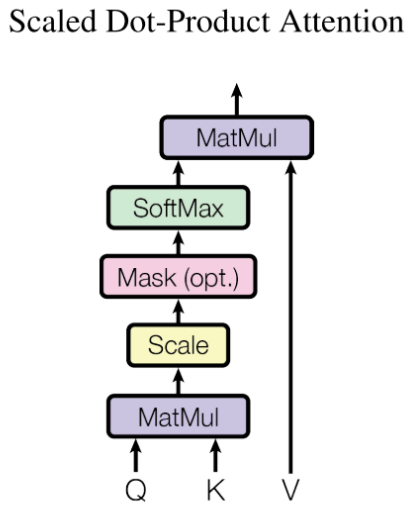


Figure 1: The Transformer - model architecture.



$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,...,\text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



$$\text{Attention}(Q,K,V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Position-wise Feed-Forward

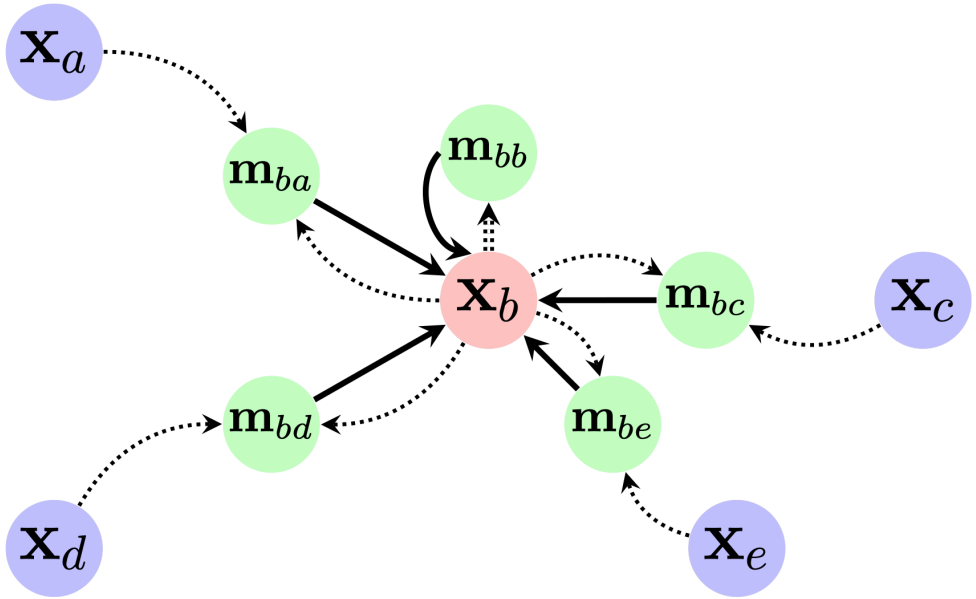
$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Positional Encoding

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_\text{model}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_\text{model}})$$

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, Illia Polosukhin. "**Attention Is All You Need**." In *NeurIPS*, 2017

Background: Message Passing GNNs (MPNNs)



Message-passing

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} \psi(\mathbf{x}_u, \mathbf{x}_v)\right)$$

Deficiencies of MPNN:

(i) High/hard inductive bias

(ii) Limited expressive power

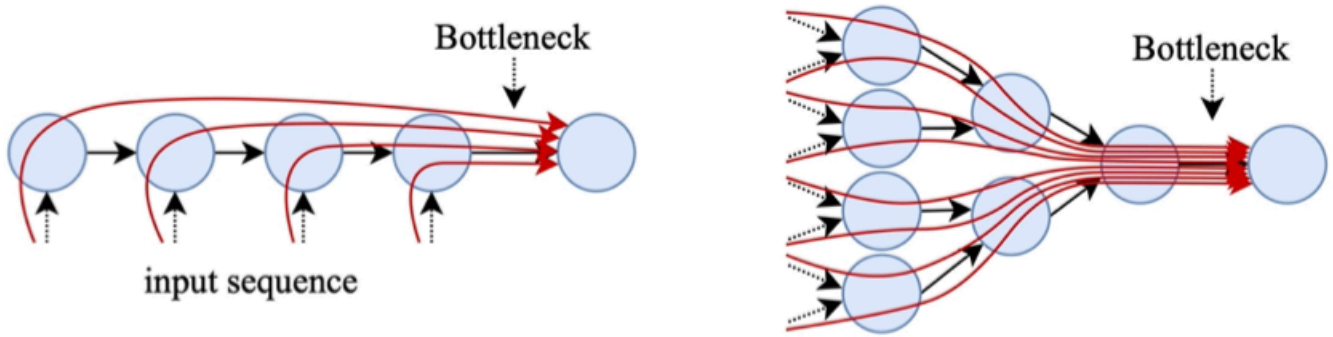
(iii) Over-smoothing and over squashing

[1] Gilmer, J. et al. **Neural Message Passing for Quantum Chemistry**. ICML, 2017

[2] Veličković, P. **Message passing all the way up**. arXiv , 2022

Over-squashing

Over-squashing: exponential blow-up in computation paths as the model depth increases



(a) The bottleneck of RNN seq2seq models

(b) The bottleneck of graph neural networks

Figure 1: The bottleneck that existed in RNN seq2seq models (before attention) is strictly more harmful in GNNs: information from a node's exponentially-growing receptive field is compressed into a fixed-size vector. Black arrows are graph edges; red curved arrows illustrate information flow.

GNNs fail to propagate messages originating from distant nodes and perform poorly when the prediction task depends on long-range interaction.

[Alon et al., 2021] On the Bottleneck of Graph Neural Networks and its Practical Implications. ICLR 2021.

UGformer, WWW, 2022

1. MPNN
2. Transformer-based MPNN
3. Graph Transformer

2.1 Variant 1: Leveraging the transformer on a set of sampled neighbors for each node

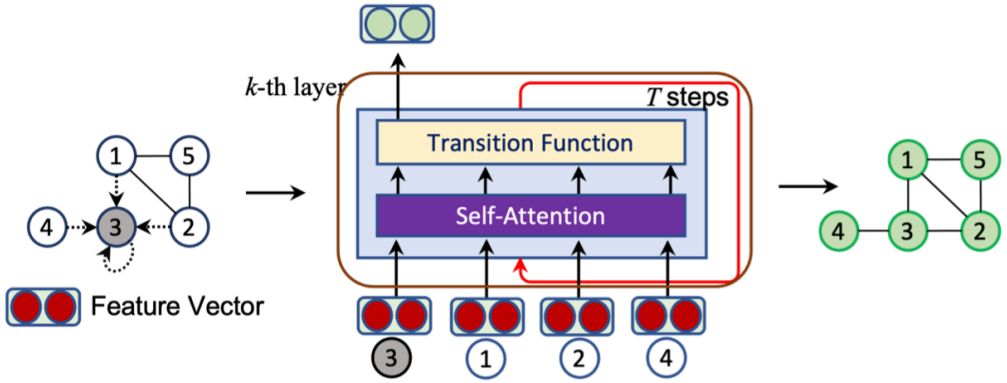


Figure 1: An illustration of UGformer Variant 1.

Transformer-based MPNN

2.2 Variant 2: Leveraging the transformer on all input nodes

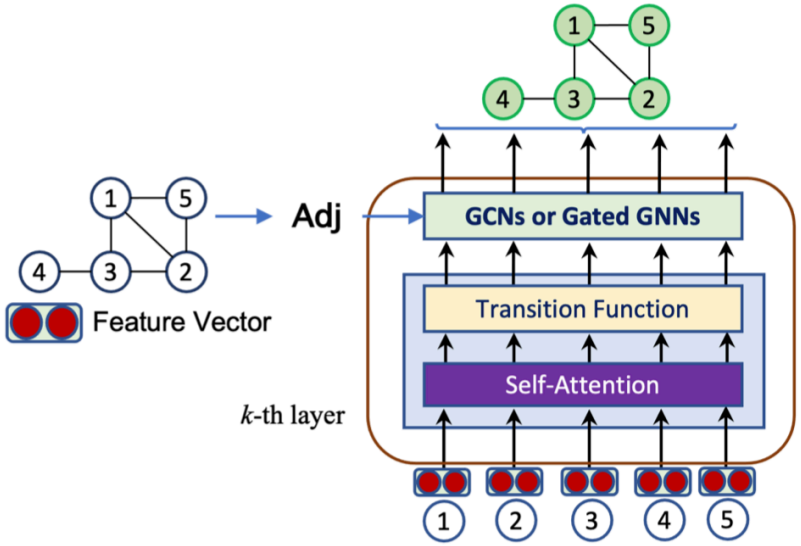


Figure 2: An illustration of UGformer Variant 2.

Graph Transformer

[1] Dai Quoc Nguyen, Tu Dinh Nguyen, Dinh Phung. "**Universal Graph Transformer Self-Attention Networks**." In *WWW Companion*. 2022

Message Passing All the Way Up, arXiv, 2022

Graph rewiring The previous categories did not require moving away from message passing over exactly the given graph—at most, constraints needed to be placed on ψ. However, as the input graph is often noisy, missing, or suboptimal for the task, many methods *modify* the edges of the input graph to compensate. Such *graph rewiring* methods leave $\mathcal{V}$ unchanged, but make direct changes to $\mathcal{N}_u$.

One common method in this space is to operate over a *fully connected graph*, i.e. setting $\mathcal{N}_u = \mathcal{V}$. This allows the model to rediscover the edges it needs, and has become quite popular in the context of *graph Transformers* (Ying et al., 2021; Kreuzer et al., 2021; Mialon et al., 2021), which allows for building GNNs that are able to "win the hardware lottery" (Hooker, 2021). Conveniently, the fully

- Why GT works
  - FC graphs + attention handles the over-squashing problem
  - Soft (low) inductive bias
  - Higher expressive power (beyond WL-test)

[1] Petar Veličković. "**Message Passing All the Way Up**." *ArXiv*, 2022

Localized Graph Transformer (Transformer-based MPNN)

Restrict self-attention to local neighborhoods

GAT, ICLR, 2018

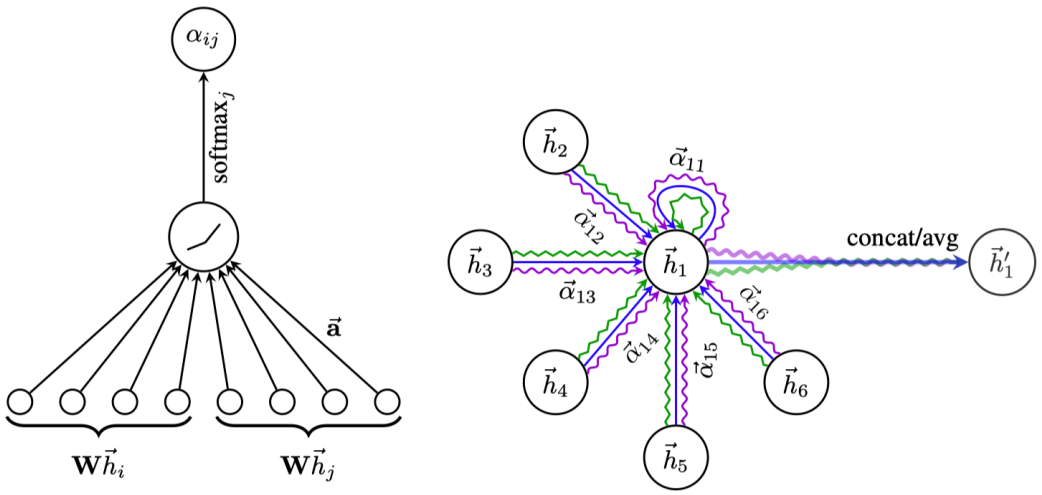


Figure 1: **Left:** The attention mechanism $a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$ employed by our model, parametrized by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$, applying a LeakyReLU activation. **Right:** An illustration of multi-head attention (with $K = 3$ heads) by node 1 on its neighborhood. Different arrow styles and colors denote independent attention computations. The aggregated features from each head are concatenated or averaged to obtain $\vec{h}'_1$.

Attention Mechanism

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$

$j \in \mathcal{N}_i$, where $\mathcal{N}_i$ is some *neighborhood* of node i in the graph.

$$\vec{h}'_i = \sigma\left(\sum_{j\in\mathcal{N}_i}\alpha_{ij}\mathbf{W}\vec{h}_j\right).$$

Multi-head Attention

$$\vec{h}'_i = \Big\Vert_{k=1}^{K}\sigma\left(\sum_{j\in\mathcal{N}_i}\alpha_{ij}^k\mathbf{W}^k\vec{h}_j\right)$$

[1] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, Yoshua Bengio. "**Graph Attention Networks**." In *ICLR*, 2018

Graph Transformer, DGL@AAAI, 2021

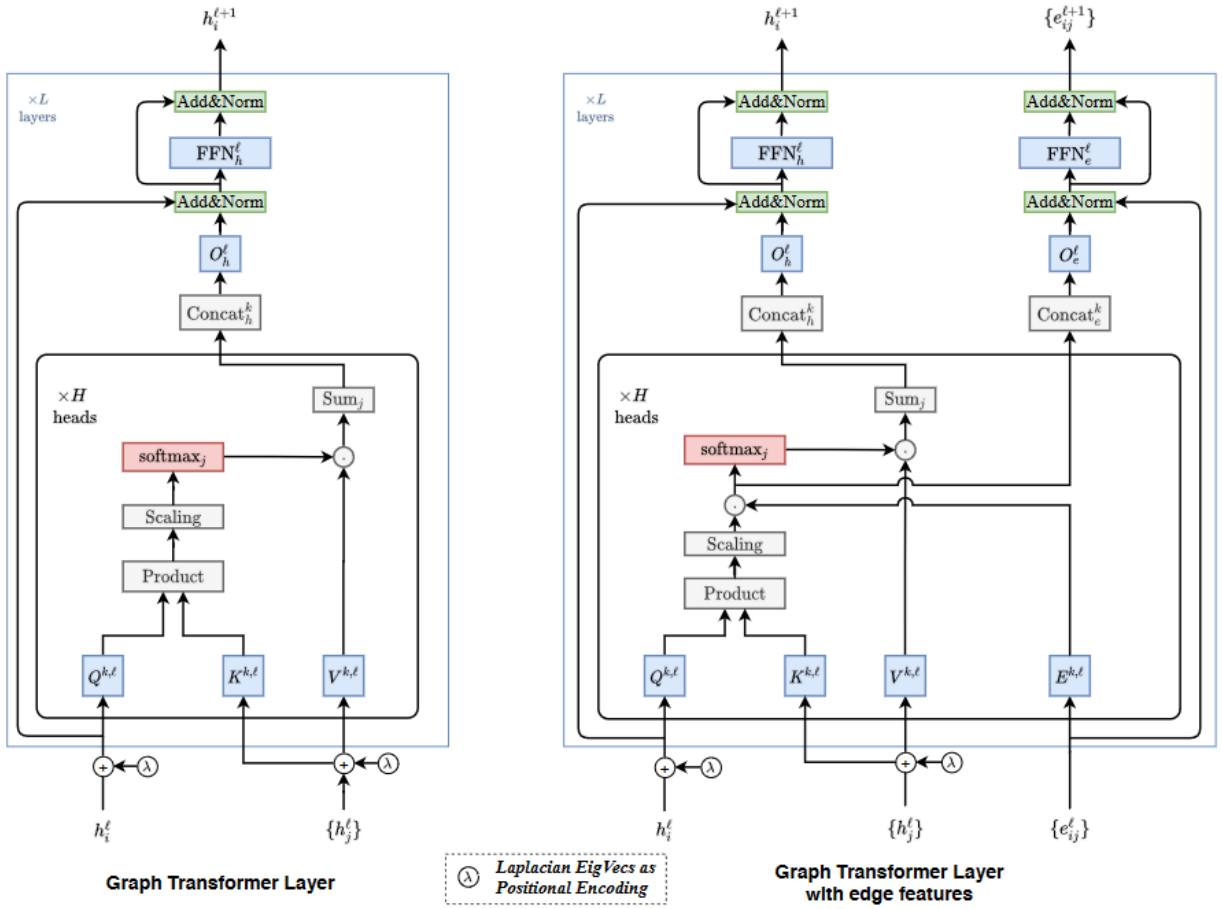


Figure 1: Block Diagram of Graph Transformer with Laplacian Eigvectors (λ) used as positional encoding (LapPE). LapPE is added to input node embeddings before passing the features to the first layer. ***Left***: Graph Transformer operating on node embeddings only to compute attention scores; ***Right***: Graph Transformer with edge features with designated feature pipeline to maintain layer wise edge representations. In this extension, the available edge attributes in a graph is used to explicitly modify the corresponding pairwise attention scores.

Graph Tansformer Layer

Self-Attention

$$\hat{h}_i^{\ell+1} = O_h^{\ell} \Big\Vert_{k=1}^{H} \Big(\sum_{j \in \mathcal{N}_i} w_{ij}^{k,\ell} V^{k,\ell} h_j^{\ell} \Big), \quad (4)$$

$$\text{where, } w_{ij}^{k,\ell} = \text{softmax}_j \Big(\frac{Q^{k,\ell} h_i^{\ell} \cdot K^{k,\ell} h_j^{\ell}}{\sqrt{d_k}} \Big), \quad (5)$$

Feed Forward

$$\hat{\hat{h}}_i^{\ell+1} = \text{Norm}\Big(h_i^{\ell} + \hat{h}_i^{\ell+1} \Big), \quad (6)$$

$$\hat{\hat{h}}_i^{\ell+1} = W_2^{\ell} \text{ReLU}(W_1^{\ell} \hat{\hat{h}}_i^{\ell+1}), \quad (7)$$

$$h_i^{\ell+1} = \text{Norm}\Big(\hat{\hat{h}}_i^{\ell+1} + \hat{\hat{h}}_i^{\ell+1} \Big), \quad (8)$$

Graph Tansformer Layer with Edge Feature

Self-Attention

$$\hat{h}_i^{\ell+1} = O_h^{\ell} \Big\Vert_{k=1}^{H} \Big(\sum_{j \in \mathcal{N}_i} w_{ij}^{k,\ell} V^{k,\ell} h_j^{\ell} \Big), \quad (9)$$

$$\hat{e}_{ij}^{\ell+1} = O_e^{\ell} \Big\Vert_{k=1}^{H} \Big(\hat{w}_{ij}^{k,\ell} \Big), \text{ where,} \quad (10)$$

$$w_{ij}^{k,\ell} = \text{softmax}_j(\hat{w}_{ij}^{k,\ell}), \quad (11)$$

$$\hat{w}_{ij}^{k,\ell} = \Big(\frac{Q^{k,\ell} h_i^{\ell} \cdot K^{k,\ell} h_j^{\ell}}{\sqrt{d_k}} \Big) \cdot E^{k,\ell} e_{ij}^{\ell}, \quad (12)$$

Feed Forward for Node Feature

$$\hat{\hat{h}}_i^{\ell+1} = \text{Norm}\Big(h_i^{\ell} + \hat{h}_i^{\ell+1} \Big), \quad (13)$$

$$\hat{\hat{h}}_i^{\ell+1} = W_{h,2}^{\ell} \text{ReLU}(W_{h,1}^{\ell} \hat{\hat{h}}_i^{\ell+1}), \quad (14)$$

$$h_i^{\ell+1} = \text{Norm}\Big(\hat{\hat{h}}_i^{\ell+1} + \hat{\hat{h}}_i^{\ell+1} \Big), \quad (15)$$

Feed Forward for Edge Feature

$$\hat{\hat{e}}_{ij}^{\ell+1} = \text{Norm}\Big(e_{ij}^{\ell} + \hat{e}_{ij}^{\ell+1} \Big), \quad (16)$$

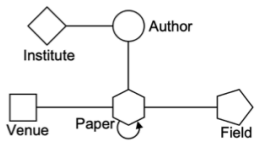
$$\hat{\hat{e}}_{ij}^{\ell+1} = W_{e,2}^{\ell} \text{ReLU}(W_{e,1}^{\ell} \hat{\hat{e}}_{ij}^{\ell+1}), \quad (17)$$

$$e_{ij}^{\ell+1} = \text{Norm}\Big(\hat{\hat{e}}_{ij}^{\ell+1} + \hat{\hat{e}}_{ij}^{\ell+1} \Big), \quad (18)$$

https://github.com/graphdeeplearning/graphtransformer

[1] Vijay Prakash Dwivedi, Xavier Bresson. "**A Generalization of Transformer Networks to Graphs**." In *DLG@AAAI*, 2021

HGT, WWW, 2020

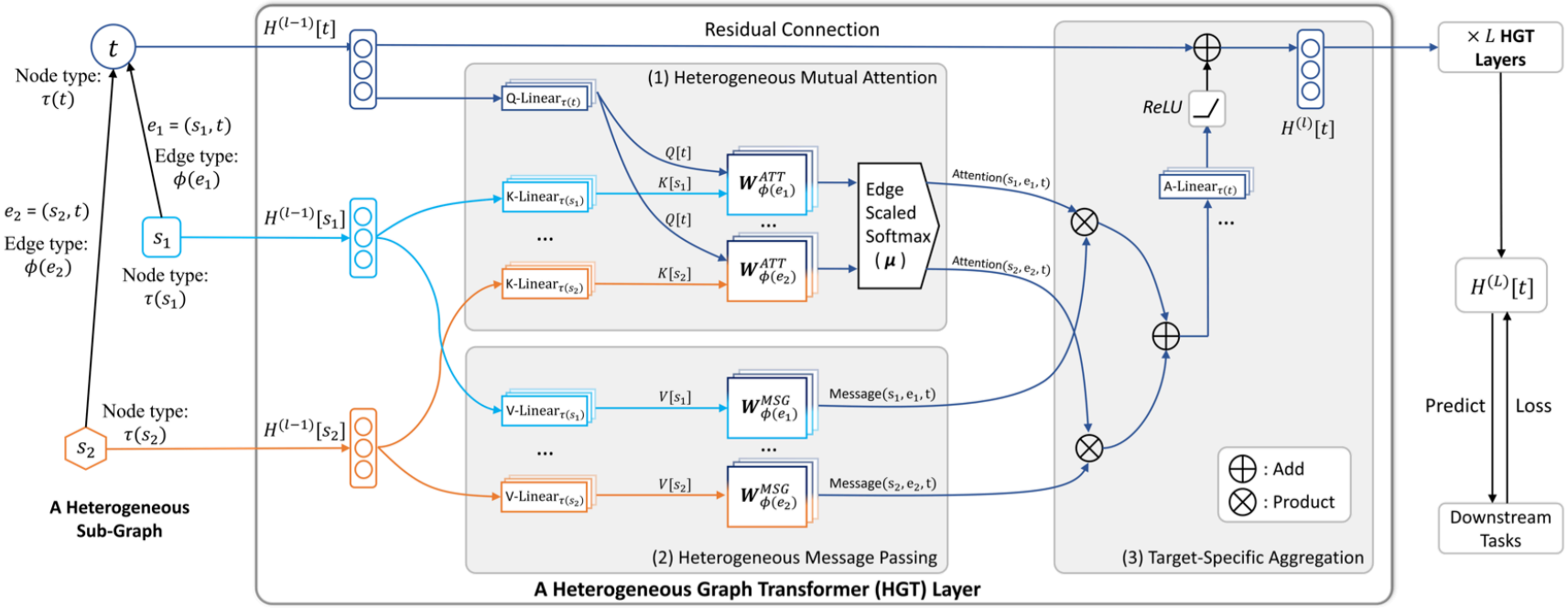


(a) The schema of heterogeneous academic networks

Author	*is_(first/last/other)_author_of*	Paper
Author	*is_affiliated_with*	Institute
Paper	*is_published_(conf/journal)_at*	Venue
Paper	*has_(L_1–L_5)_field_of*	Field
Paper	*has_citation_to*	Paper

(b) The meta relations of heterogeneous academic networks

Figure 1: The schema and meta relations of Open Academic Graph (OAG).



$$\textbf{Attention}_{HGT}(s,e,t) = \underset{\forall s \in N(t)}{\text{Softmax}}\Big(\Big\|_{i\in[1,h]} ATT\text{-}head^i(s,e,t)\Big) \quad (2)$$

$$ATT\text{-}head^i(s,e,t) = \Big(K^i(s)\, W^{ATT}_{\phi(e)}\, Q^i(t)^T\Big) \cdot \frac{\mu_{\langle \tau(s),\phi(e),\tau(t)\rangle}}{\sqrt{d}}$$

$$K^i(s) = \text{K-Linear}^i_{\tau(s)}\Big(H^{(l-1)}[s]\Big)$$

$$Q^i(t) = \text{Q-Linear}^i_{\tau(t)}\Big(H^{(l-1)}[t]\Big)$$

Figure 2: The Overall Architecture of Heterogeneous Graph Transformer. Given a sampled heterogeneous sub-graph with t as the target node, s_1 & s_2 as source nodes, the HGT model takes its edges $e_1 = (s_1, t)$ & $e_2 = (s_2, t)$ and their corresponding meta relations $< \tau(s_1), \phi(e_1), \tau(t) >$ & $< \tau(s_2), \phi(e_2), \tau(t) >$ as input to learn a contextualized representation $H^{(L)}$ for each node, which can be used for downstream tasks. Color decodes the node type. HGT includes three components: (1) meta relation-aware heterogeneous mutual attention, (2) heterogeneous message passing from source nodes, and (3) target-specific heterogeneous message aggregation.

[1] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Yizhou Sun. "**Heterogeneous Graph Transformer**." In *WWW*, 2020

MitiGaTe, NAACL, 2022

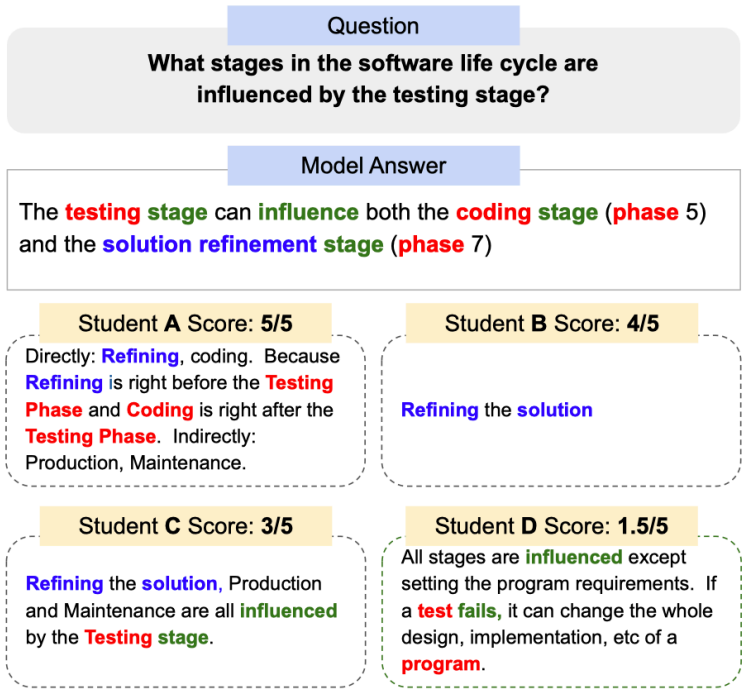


Figure 1: A motivating example for using multiple relations in automatic short answer grading.

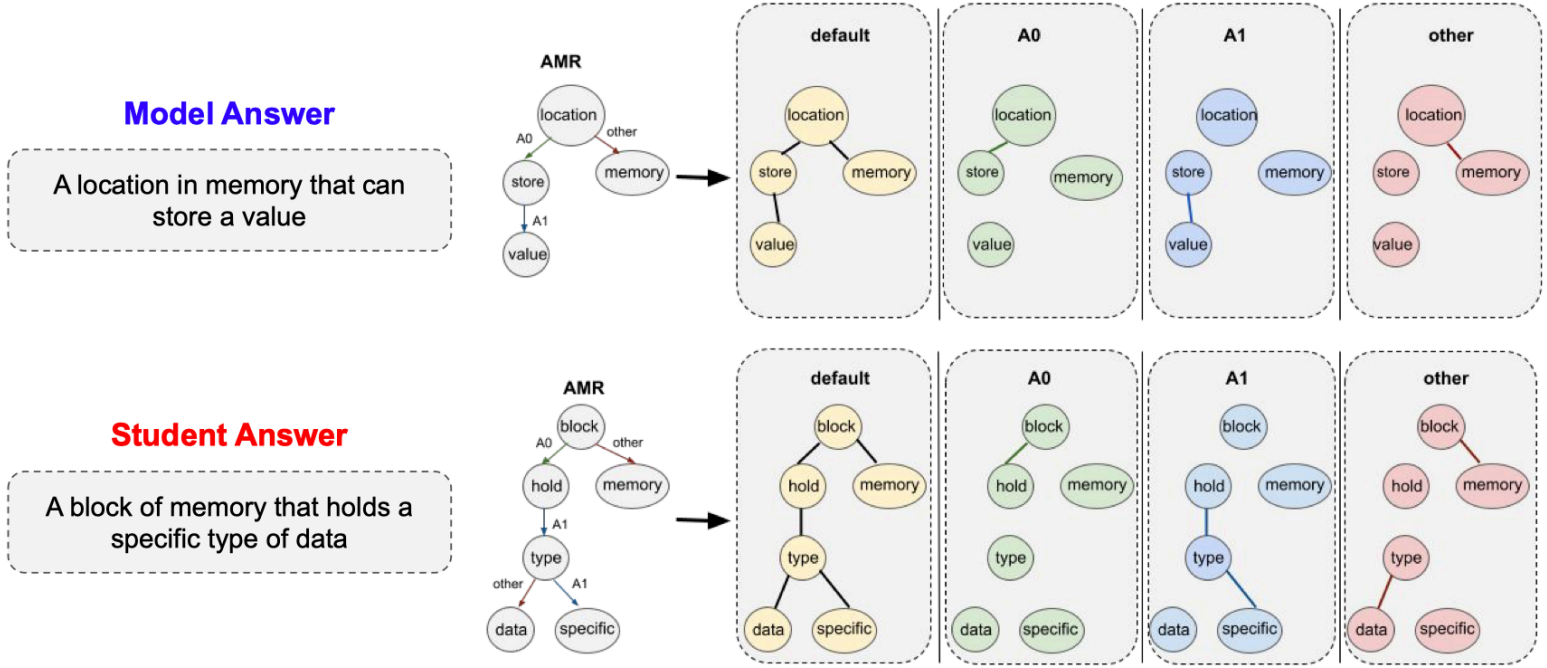


Figure 2: Example to show subgraph splitting and intuition behind using AMR for ASAG task. Similar relations in model and student answers have similar structures. Example, *location* and *store* in model answer are connected as A_0 and similarly *block* and *hold* in student answer are connected with A_0. This applies to other relations as well.

[1] Rajat Agarwal, Varun Khurana, Karish Grover, Mukesh Mohania, Vikram Goyal. "**Multi-Relational Graph Transformer for Automatic Short Answer Grading**." In *NAACL*. 2022

MitiGaTe, NAACL, 2022

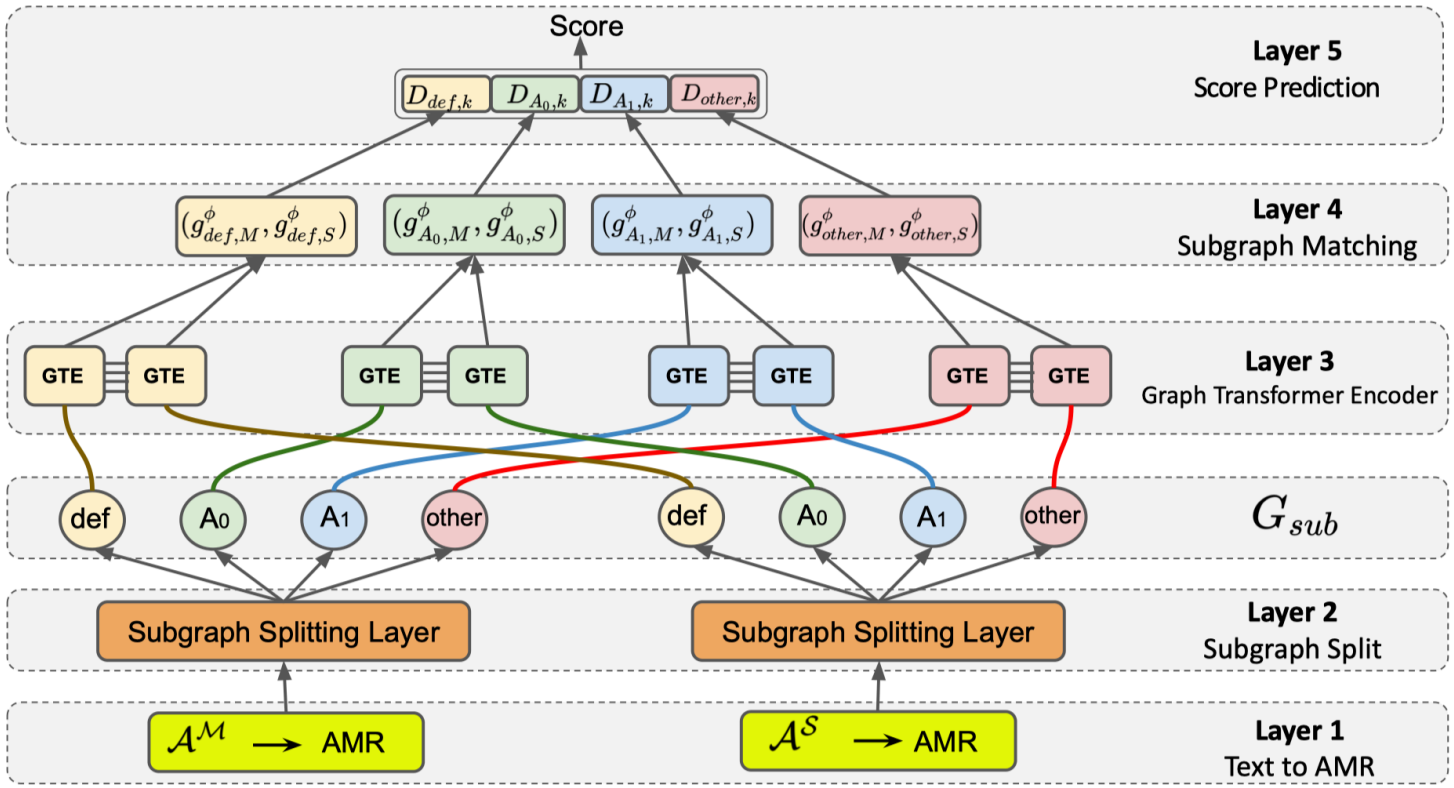


Figure 3: The framework of the proposed model **MitiGaTe** employs Graph Transformer for learning the structural context of a sentence. Please refer to Section 3 for more details.

$$h_i^{l+1} = O_h^l ||_{k=1}^{H} \Big(\sum_{j \in \mathcal{N}_i^+} w_{ij}^{k,l} V^{k,l} h_j^l \Big) \quad (2)$$

$$e_{ij}^{l+1} = O_e^l ||_{k=1}^{H} (\hat{w}_{ij}^{k,l}) \quad (3)$$

$$w_{ij}^{k,l} = \text{softmax}_j(\hat{w}_{ij}^{k,l}) \quad (4)$$

$$\hat{w}_{ij}^{k,l} = \Big(\frac{Q^{k,l} h_i^l \cdot K^{k,l} h_j^l}{\sqrt{d_k}} \Big) \cdot E^{k,l} e_{ij}^l \quad (5)$$

$\mathcal{N}_i^+$ is the set of neighbors of node i in the graph including self-loop.

[1] Rajat Agarwal, Varun Khurana, Karish Grover, Mukesh Mohania, Vikram Goyal. "**Multi-Relational Graph Transformer for Automatic Short Answer Grading**." In *NAACL*. 2022

Global Graph Transformer

self-attention on entire graph

- Structural/Positional Encoding
- Scaling to Large Graphs

Graph Transformer + PE

Absolute PE: added into node embedding

Relative PE: injected into global self-attention via attention bias

Relative PE, NAACL, 2018

Canonical self-attention

$$z_i = \sum_{j=1}^{n} \alpha_{ij}(x_j W^V) \qquad (1)$$

$$\alpha_{ij} = \frac{\exp e_{ij}}{\sum_{k=1}^{n} \exp e_{ik}}$$

$$e_{ij} = \frac{(x_i W^Q)(x_j W^K)^T}{\sqrt{d_z}} \qquad (2)$$

introduce relative PE

The edge between input elements x_i and x_j is represented by vectors $a_{ij}^V, a_{ij}^K \in \mathbb{R}^{d_a}$. The mo-

$$z_i = \sum_{j=1}^{n} \alpha_{ij}(x_j W^V + a_{ij}^V) \qquad (3)$$

$$\alpha_{ij} = \frac{\exp e_{ij}}{\sum_{k=1}^{n} \exp e_{ik}}$$

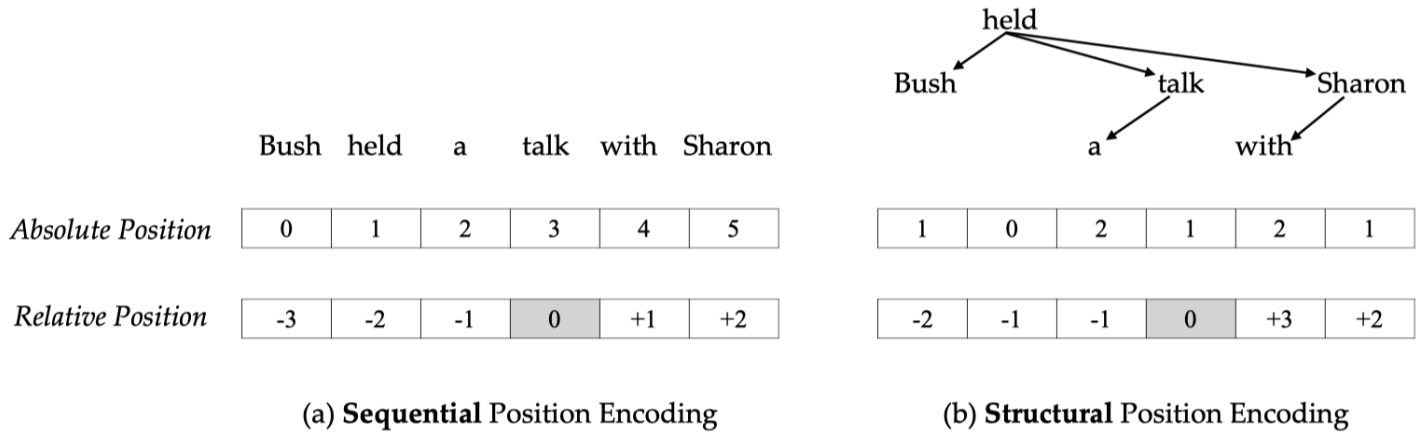
$$e_{ij} = \frac{x_i W^Q (x_j W^K + a_{ij}^K)^T}{\sqrt{d_z}} \qquad (4)$$

Efficient implementation

$$e_{ij} = \frac{x_i W^Q (x_j W^K)^T + x_i W^Q (a_{ij}^K)^T}{\sqrt{d_z}} \qquad (5)$$

[1] Peter Shaw, Jakob Uszkoreit, Ashish Vaswani. "**Self-Attention with Relative Position Representations**." In *NAACL*. 2018

Structural PE, EMNLP, 2019



(a) **Sequential** Position Encoding (b) **Structural** Position Encoding

Figure 1: Illustration of (a) the standard *sequential position encoding* (Vaswani et al., 2017; Shaw et al., 2018), and (b) the proposed *structural position encoding*. The relative position in the example is for the word "talk".

Absolute Structural PE

$$abs_{stru}(x_i) = distance_{tree}(x_i, origin), \quad (5)$$

Relative Structural PE

1. if x_i and x_j are at same dependency edge, $rel_{stru}(x_i, x_j) = abs_{stru}(x_i) - abs_{stru}(x_j)$.
2. if x_i and x_j are at different dependency edges, $rel_{stru}(x_i, x_j) = f_{stru}(i - j) * (abs_{stru}(x_i) + abs_{stru}(x_j))$, where

$$f_{stru}(x) = \begin{cases} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases} \quad (6)$$

[1] Xing Wang, Zhaopeng Tu, Longyue Wang, Shuming Shi. "**Self-Attention with Structural Position Representations**." In *EMNLP/IJCNLP*. 2019

Graphormer, NeurIPS, 2021

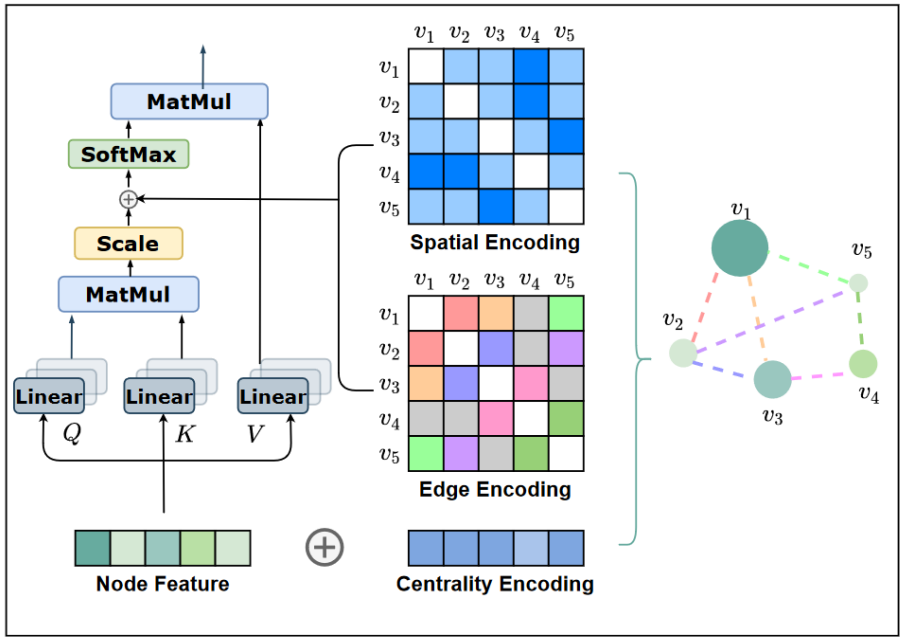


Figure 1: An illustration of our proposed centrality encoding, spatial encoding, and edge encoding in Graphormer.

https://github.com/Microsoft/Graphormer

Self-Attention

$$Q = HW_Q, \quad K = HW_K, \quad V = HW_V, \tag{3}$$

$$A = \frac{QK^\top}{\sqrt{d_K}}, \quad \text{Attn}\,(H) = \text{softmax}\,(A)\,V, \tag{4}$$

Centrality Encoding

$$h_i^{(0)} = x_i + z^-_{\deg^-(v_i)} + z^+_{\deg^+(v_i)}, \tag{5}$$

Spatial Encoding (shortest path distance)

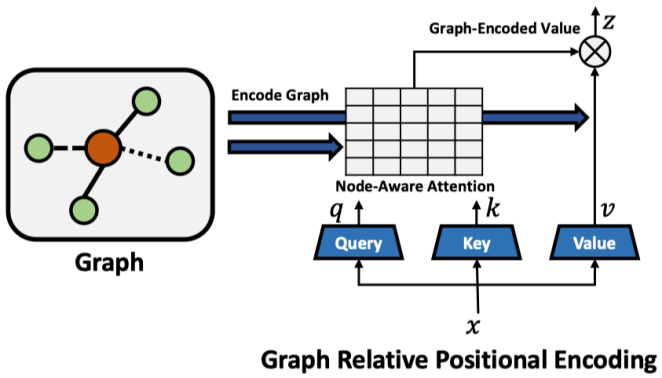
$$A_{ij} = \frac{(h_i W_Q)(h_j W_K)^T}{\sqrt{d}} + b_{\phi(v_i, v_j)}, \tag{6}$$

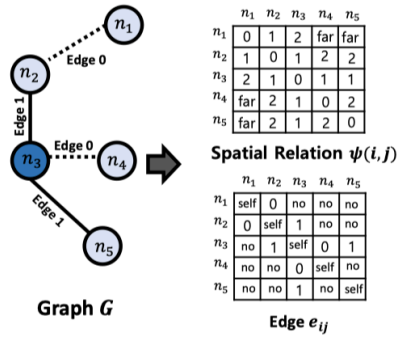
Edge Encoding

$$A_{ij} = \frac{(h_i W_Q)(h_j W_K)^T}{\sqrt{d}} + b_{\phi(v_i, v_j)} + c_{ij}, \text{ where } c_{ij} = \frac{1}{N}\sum_{n=1}^{N} x_{e_n}(w_n^E)^T, \tag{7}$$

[1] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, Tie-Yan Liu. "**Do Transformers Really Perform Bad for Graph Representation?**" In *NeurIPS*, 2021

GRPE, MLDD@ICLR, 2022





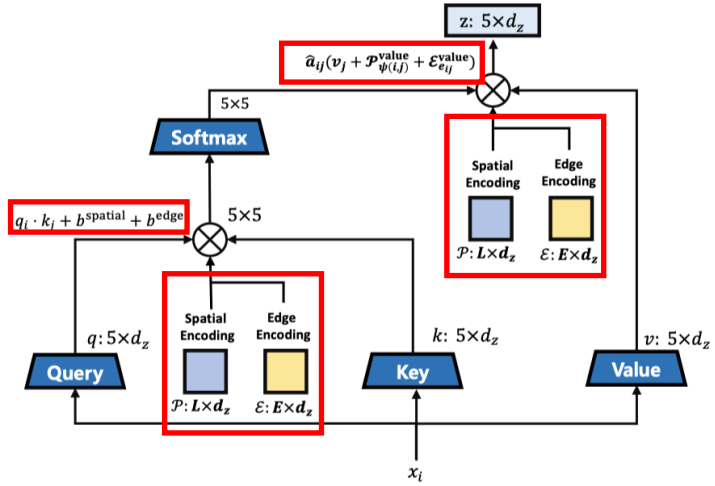


Figure 2: Illustration of the proposed Graph Relative Positional Encoding. Left figure shows an example of how GRPE process relative relation between nodes. In the example we set the L to 2. Right figure describes our self-attention mechanism. Our two relative positional encodings, spatial encoding and edge encoding, are used to encode graph on both attention map and value.

$$b_{ij}^{\text{spatial}} = q_i \cdot \mathcal{P}_{\psi(i,j)}^{\text{query}} + k_j \cdot \mathcal{P}_{\psi(i,j)}^{\text{key}} \tag{6}$$

$$b_{ij}^{\text{edge}} = q_i \cdot \mathcal{E}_{e_{ij}}^{\text{query}} + k_j \cdot \mathcal{E}_{e_{ij}}^{\text{key}} \tag{7}$$

$$a_{ij} = \frac{q_i \cdot k_j + b_{ij}^{\text{spatial}} + b_{ij}^{\text{edge}}}{\sqrt{d_z}}. \tag{8}$$

$$z_i = \sum_{j=1}^{N} \hat{a}_{ij}(v_j + \mathcal{P}_{\psi(i,j)}^{\text{value}} + \mathcal{E}_{e_{ij}}^{\text{value}}). \tag{9}$$

[1] Wonpyo Park, Woonggi Chang, Donggeon Lee, Juntae Kim, Seung-won Hwang. "**GRPE: Relative Positional Encoding for Graph Transformer**." In *MLDD@ICLR*, 2022

EGT, KDD, 2022

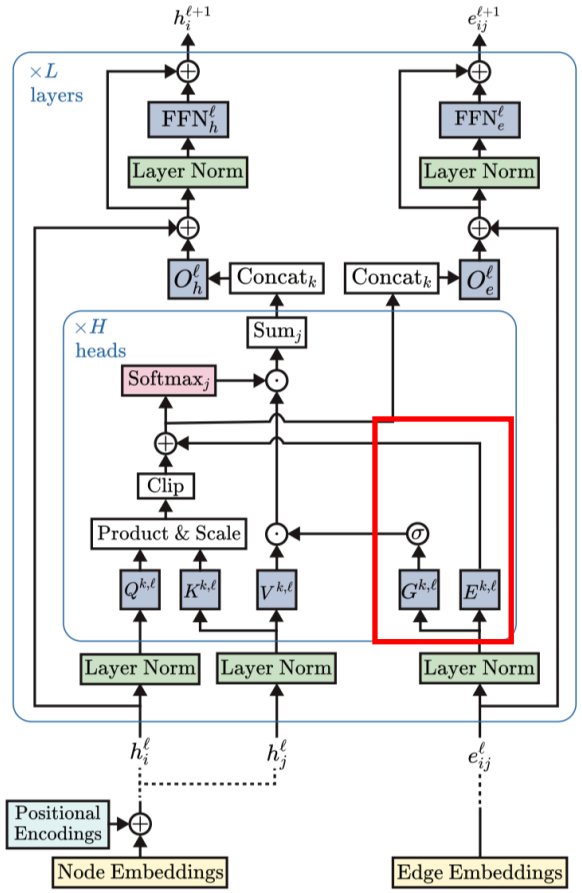


Figure 2: EGT: Edge-augmented Graph Transformer

$$\tilde{\mathbf{A}}^{k,\ell} = \text{softmax}(\hat{\mathbf{H}}^{k,\ell}) \odot \sigma(\mathbf{G}_e^{k,\ell}) \quad (4)$$

$$\text{Where,} \quad \hat{\mathbf{H}}^{k,\ell} = \text{clip}(\hat{\mathbf{A}}^{k,\ell}) + \mathbf{E}_e^{k,\ell} \quad (5)$$

where $\odot$ denotes elementwise product. $\mathbf{E}_e^{k,\ell}, \mathbf{G}_e^{k,\ell} \in \mathbb{R}^{N \times N}$ are concatenations of the learned linear transformed edge embeddings, i.e., $\mathbf{E}^{k,\ell}\hat{e}_{ij}^\ell, \mathbf{G}^{k,\ell}\hat{e}_{ij}^\ell$ respectively. $\mathbf{E}_e^{k,\ell}$ is a bias term added to the scaled dot product. It lets the edge channels influence the attention process. $\mathbf{G}_e^{k,\ell}$ drives the sigmoid $\sigma(\cdot)$ function and lets the edge channels also gate the values before aggregation. The scaled dot product is clipped to a limited range which leads to better numerical stability. We used a range of $[-5, +5]$.

~~[1] Md Shamim Hussain, Mohammed J. Zaki, Dharmashankar Subramanian. "**Edge-Augmented Graph Transformers: Global Self-Attention Is Enough for Graphs**." *ArXiv*, 2021~~

[1] Md Shamim Hussain, Mohammed J. Zaki, Dharmashankar Subramanian. "Global Self-Attention as a Replacement for Graph Convolution." In *KDD*, 2022

GraphiT, ArXiv, 2021

Transformer Architectures for Graphs

$$\text{Attention}(Q,V) = \text{softmax}\left(\frac{QQ^\top}{\sqrt{d_{out}}}\right)V \in \mathbb{R}^{n\times d_{\text{out}}}, \tag{4}$$

$$X = X + \text{Attention}(Q,V).$$

Kernels on Graphs

$$K_r = \sum_{i=1}^{m} r(\lambda_i) u_i u_i^\top, \tag{1}$$

Diffusion Kernel

$$K_D = \sum_{i=1}^{m} e^{-\beta\lambda_i} u_i u_i^\top = e^{-\beta L} = \lim_{p\to+\infty}\left(I - \frac{\beta}{p}L\right)^p. \tag{2}$$

p-step random walk kernel

$$K_{pRW} = (I - \gamma L)^p. \tag{3}$$

Encoding Node Positions

Absolute positional encoding for graphs
graph Laplacian (LapPE)

Relative position encoding strategies by using Kernels on Graphs

$$\text{PosAttention}(Q,V,K_r)_i = \sum_{j=1}^{n} \frac{\exp\left(Q_i Q_j^\top/\sqrt{d_{out}}\right)\times K_r(i,j)}{\sum_{j'=1}^{n}\exp\left(Q_i Q_{j'}^\top/\sqrt{d_{out}}\right)\times K_r(i,j')} V_j \in \mathbb{R}^{d_{\text{out}}},$$

$$X = X + D^{-\frac{1}{2}}\text{PosAttention}(Q,V,K_r), \tag{6}$$

Encoding Topological Structures

Graph convolutional kernel networks (GCKN)

Formally, let us consider a graph G with n nodes, and let us denote by $\mathcal{P}_k(u)$ the set of paths shorter than or equal to k that start with node u. With an abuse of notation, p in $\mathcal{P}_k(u)$ will denote the concatenation of all node features encountered along the path. Then, a layer of GCKN defines a feature map X in $\mathbb{R}^{n\times d}$ such that

$$X(u) = \sum_{p\in\mathcal{P}_k(u)} \psi(p),$$

https://github.com/inria-thoth/GraphiT

[1] Grégoire Mialon, Dexiong Chen, Margot Selosse, Julien Mairal. "**GraphiT: Encoding Graph Structure in Transformers**." *ArXiv*, 2021

SAN, NeurIPS, 2021

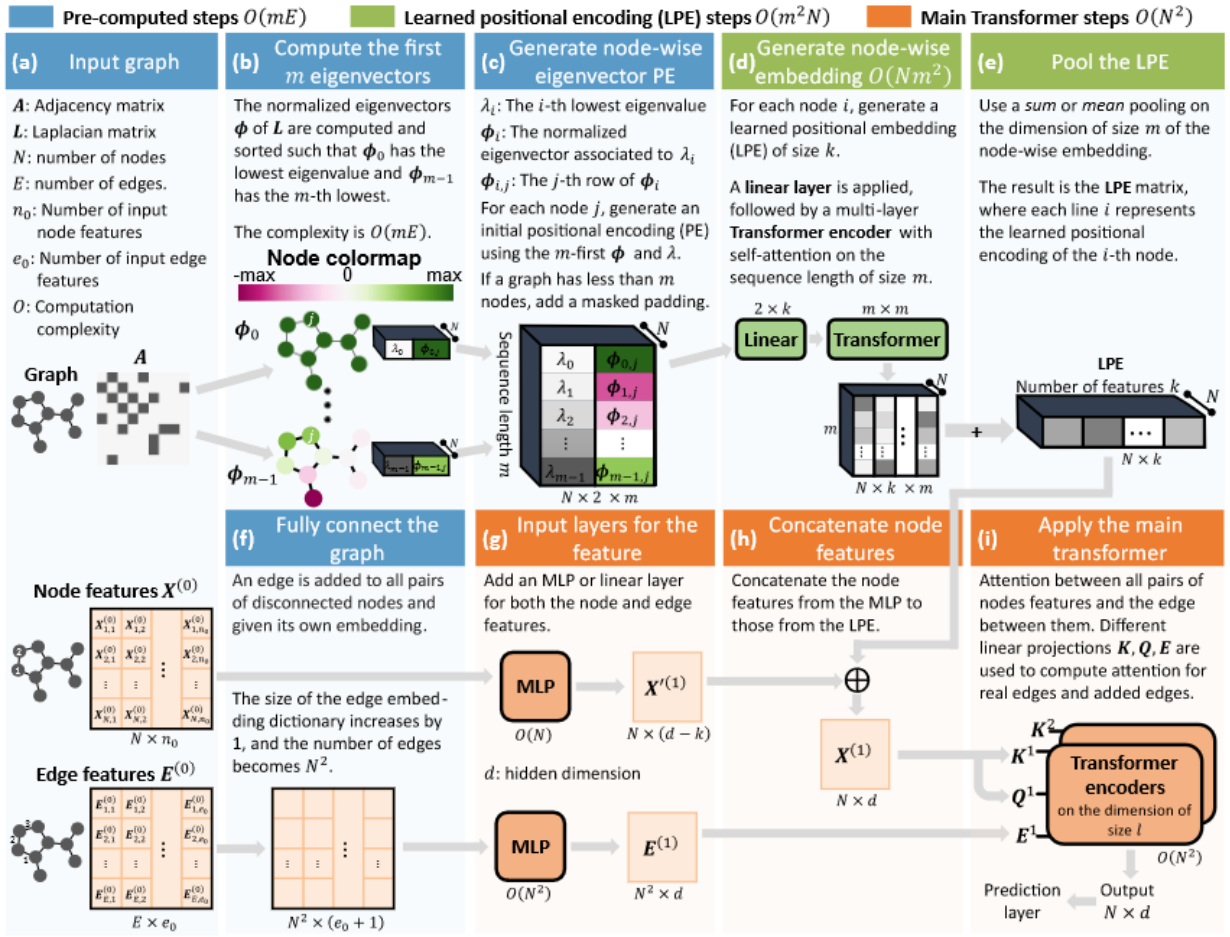


Figure 1: The proposed SAN model with the node LPE, a generalization of Transformers to graphs.

https://github.com/DevinKreuzer/SAN

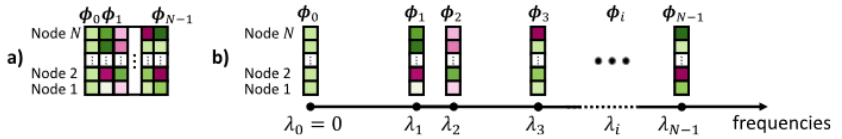


Figure 2: a) Standard view of the eigenvectors as a matrix. b) Eigenvectors ϕ_i viewed as vectors positionned on the axis of frequencies (eigenvalues).

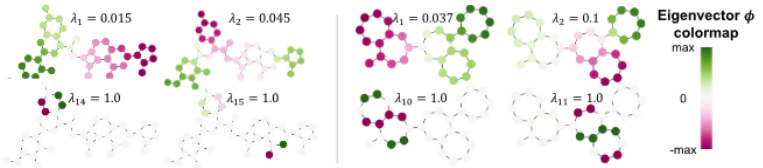


Figure 3: Examples of eigenvalues λ_i and eigenvectors ϕ_i for molecular graphs. The low-frequency eigenvectors ϕ_1, ϕ_2 are spread accross the graph, while higher frequencies, such as ϕ_{14}, ϕ_{15} for the left molecule or ϕ_{10}, ϕ_{11} for the right molecule, often resonate in local structures.

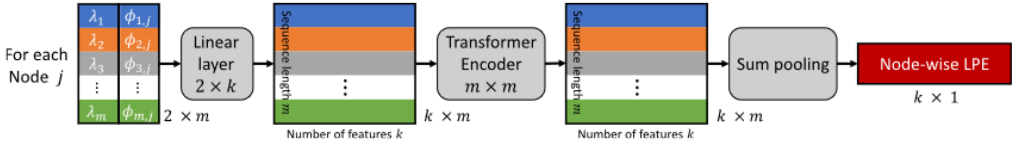


Figure 4: Learned positional encoding (LPE) architectures, with the model being aware of the graph's Laplace spectrum by considering m eigenvalues and eigenvectors, where we permit $m \leq N$, with N denoting the number of nodes. Since the Transformer loops over the nodes, each node can be viewed as an element of a batch to parallelize the computation. Here $\phi_{i,j}$ is the j-th element of the eigenvector paired to the i-th lowest eigenvalue λ_i.

$$\hat{h}_i^{l+1} = O_h^l \Big\|_{k=1}^{H} \Big(\sum_{j \in V} w_{ij}^{k,l} V^{k,l} h_j^l \Big) \tag{3}$$

$$\hat{w}_{ij}^{k,l} = \begin{cases} \dfrac{Q^{1,k,l} h_i^l \circ K^{1,k,l} h_j^l \circ E^{1,k,l} e_{ij}}{\sqrt{d_k}} & \text{if } i \text{ and } j \text{ are connected in sparse graph} \\ \dfrac{Q^{2,k,l} h_i^l \circ K^{2,k,l} h_j^l \circ E^{2,k,l} e_{ij}}{\sqrt{d_k}} & \text{otherwise} \end{cases} \tag{4}$$

$$w_{ij}^{k,l} = \begin{cases} \frac{1}{1+\gamma} \cdot \text{softmax}(\sum_{d_k} \hat{w}_{ij}^{k,l}) & \text{if } i \text{ and } j \text{ are connected in sparse graph} \\ \frac{\gamma}{1+\gamma} \cdot \text{softmax}(\sum_{d_k} \hat{w}_{ij}^{k,l}) & \text{otherwise} \end{cases} \tag{5}$$

where $\circ$ denotes element-wise multiplication

$$\hat{\hat{h}}^{l+1} = \text{Norm}(h_i^l + \hat{h}_i^{l+1}), \quad \hat{\hat{h}}_i^{l+1} = W_2^l \text{ReLU}(W_1^l \hat{\hat{h}}_i^{l+1}), \quad h_i^{l+1} = \text{Norm}(\hat{\hat{h}}^{l+1} + \hat{\hat{h}}_i^{l+1}) \tag{6}$$

[1] Devin Kreuzer, Dominique Beaini, William L. Hamilton, Vincent Létourneau, Prudencio Tossou. "**Rethinking Graph Transformers with Spectral Attention**." In *NeurIPS*, 2021

Graph Transformer + MPNN

Using global self-attention in conjunction with message-passing GNN

GTransformer / GROVER, NeurIPS, 2020

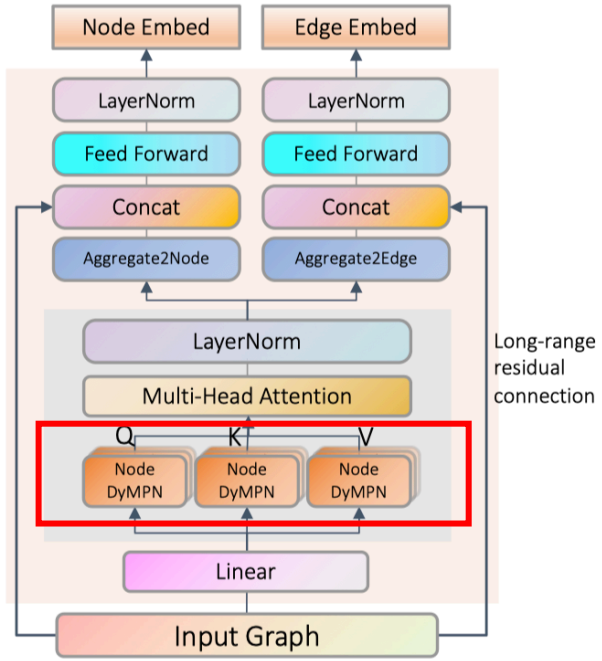


Figure 1: Overview of GTransformer.

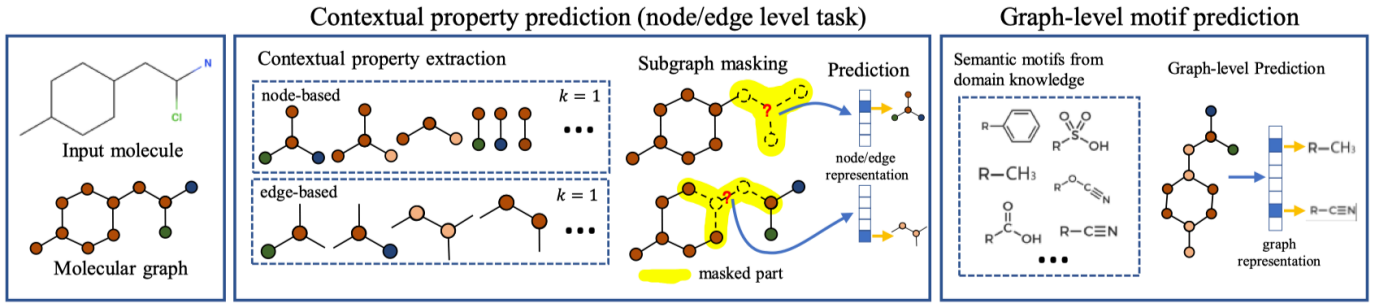


Figure 2: Overview of the designed self-supervised tasks of GROVER.

Bi-level information extraction:

- 1st level (local): MPNN captures local structural information.
- 2nd level (global): Transformer extracts global relations between nodes.

[1] Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying WEI, Wenbing Huang, Junzhou Huang. "**Self-Supervised Graph Transformer on Large-Scale Molecular Data**." In *NeurIPS*. 2020

Mesh Graphormer, ICCV, 2021

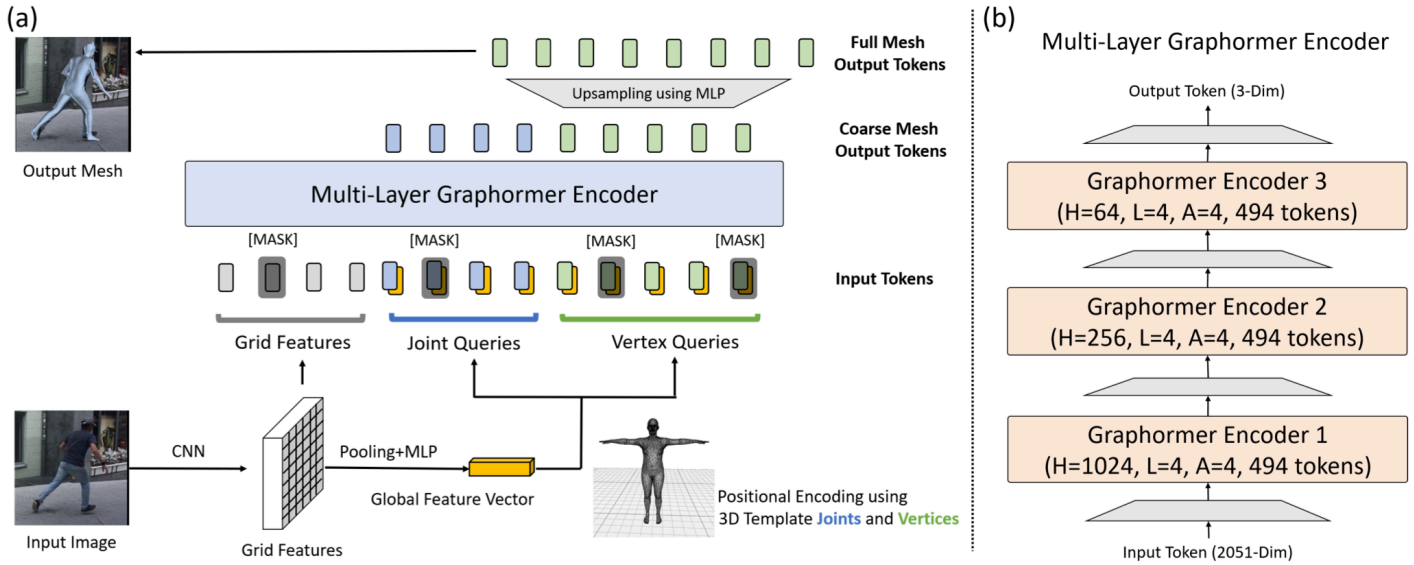


Figure 3: Graphormer for Human Mesh Reconstruction. (a) Our end-to-end mesh regression framework takes an image as input and predicts 3D joints and mesh vertices at the same time. We extract both the grid features and a global feature vector using a pre-trained CNN. The features are then tokenized and fed to a multi-layer Graphormer encoder. Grid features and multi-layer Graphormer encoder are mutually reinforced to reconstruct an accurate human mesh. (b) Our multi-layer Graphormer encoder consists of three Graphormer encoder blocks. All encoders have the same number of input tokens. We gradually reduce the hidden sizes, and output the 3D coordinates of body joints and mesh vertices.

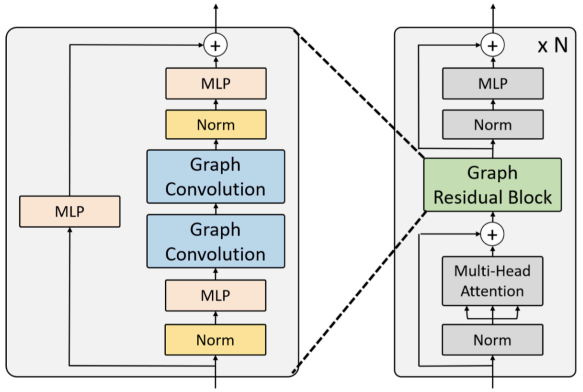
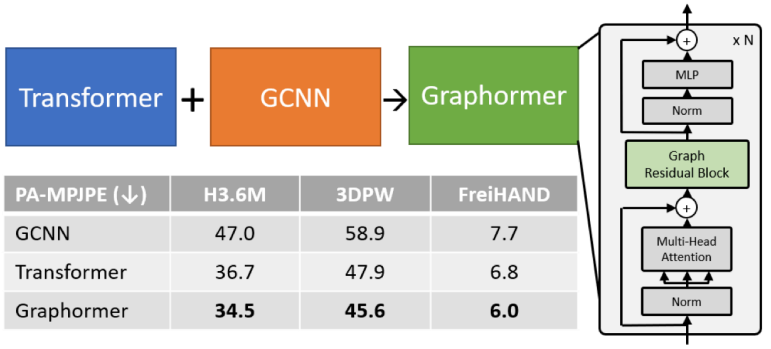


Figure 2: Architecture of a Graphormer Encoder. We propose a graph-convolution-reinforced transformer encoder to capture both global and local interactions for 3D human mesh reconstruction. The encoder consists of a stack of $N = 4$ identical blocks.



PA-MPJPE (↓)	H3.6M	3DPW	FreiHAND
GCNN	47.0	58.9	7.7
Transformer	36.7	47.9	6.8
Graphormer	**34.5**	**45.6**	**6.0**

Figure 1: Summary. We study how to combine self-attentions and graph convolutions in a transformer for human mesh reconstruction. The proposed Graphormer outperforms existing graph convolution networks [5, 20] and transformer-based method [22] by a clear margin. The numbers are the reconstruction error (PA-MPJPE) in the unit of millimeter. The lower the better.

[1] Kevin Lin, Lijuan Wang, Zicheng Liu. "**Mesh Graphormer**." In *ICCV*, 2021

GraphTrans, NeurIPS, 2021

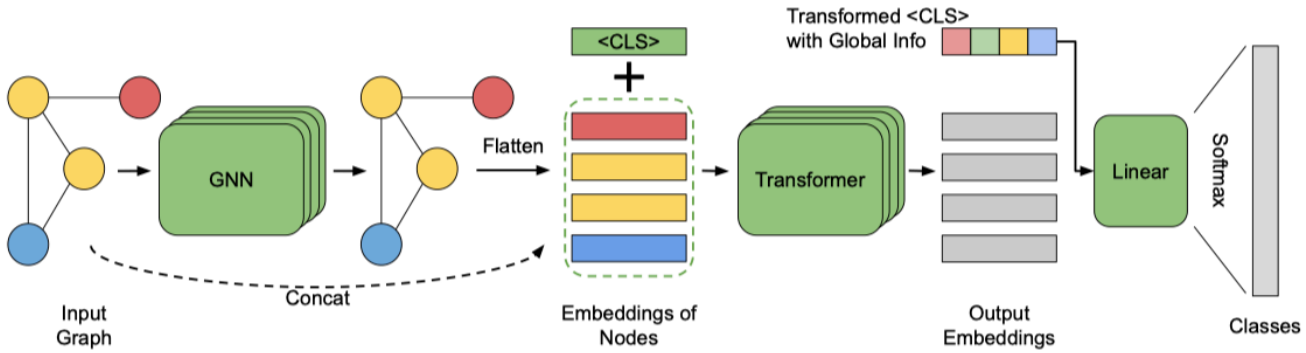
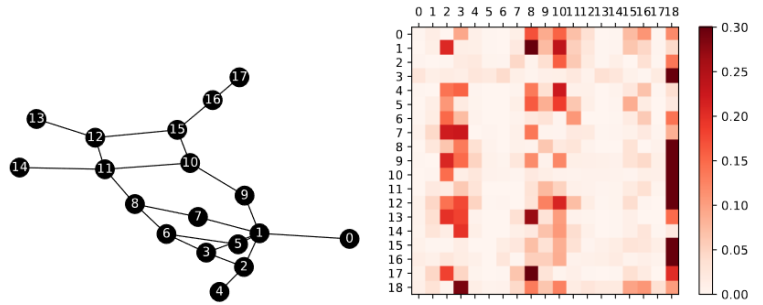


Figure 1: Architecture of `GraphTrans`. A standard GNN submodule learns local, short-range structure, then a global Transformer submodule learns global, long-range relationships.



(a) Example graph from Code2. (b) Corresponding attention map from `GraphTrans`.

Figure 2: Example graph and attention map in our `GraphTrans`. The graph is randomly sampled from the Code2 validation set. The attention map is retrieved from the first layer of the transformer module in our `GraphTrans`. The horizontal axis corresponds to targets and the horizontal axis corresponds to sources (so, attention weights will sum to one over the horizontal axis). Note that in (b), index 18 corresponds to the special <CLS> token described in section 4.

https://github.com/ucbrise/graphtrans

GNN module

$$h_v^\ell = f_\ell\left(h_v^{\ell-1}, \{h_u^{\ell-1} | u \in \mathcal{N}(v)\}\right), \quad \ell = 1, \ldots, L_{\text{GNN}}$$

Transformer module

$$\bar{h}_v^0 = \text{LayerNorm}(W^{\text{Proj}} h_v^{L_{\text{GNN}}})$$

$$a_{v,u}^\ell = (W_\ell^Q \bar{h}_v^{\ell-1})^\top (W_\ell^K \bar{h}_u^{\ell-1}) / \sqrt{d_{\text{TF}}} \qquad \alpha_{v,u}^\ell = \underset{w \in \mathcal{V}}{\text{softmax}}(a_{v,w}^\ell)$$

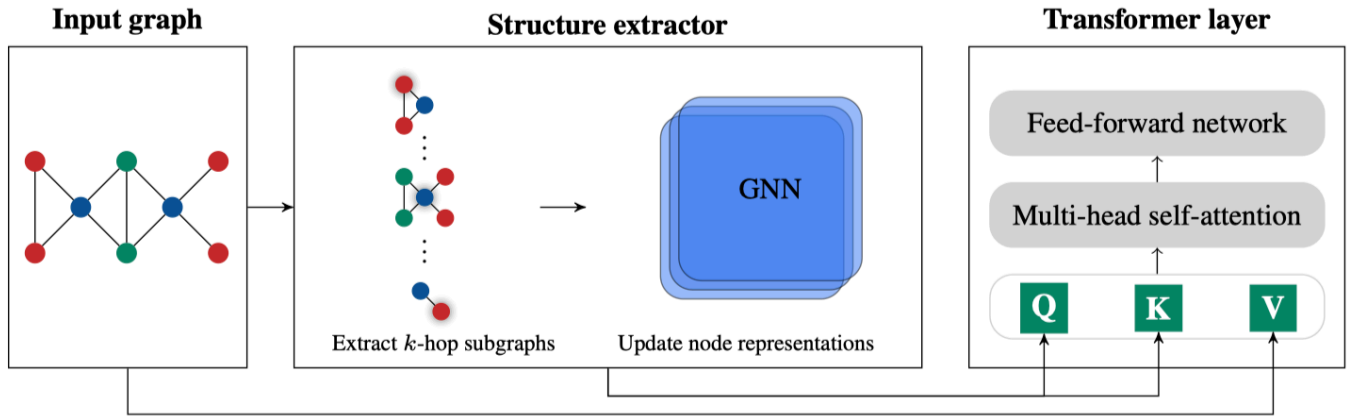
$$\bar{h}_v'^\ell = \sum_{w \in \mathcal{V}} \alpha_{v,w}^\ell W_\ell^V \bar{h}_w^{\ell-1}$$

<CLS> embedding as a GNN "readout" method

$$y = \text{softmax}(W^{\text{out}} \bar{h}_{\text{<CLS>}}^{L_{\text{TF}}}).$$

[1] Zhanghao Wu, Paras Jain, Matthew A. Wright, Azalia Mirhoseini, Joseph E. Gonzalez, Ion Stoica. "**Representing Long-Range Context for Graph Neural Networks with Global Attention**." In *NeurIPS*, 2021

SAT, ICML, 2022



Self-attention

$$\mathrm{Attn}(\mathbf{X}) := \mathrm{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_{out}}})\mathbf{V} \in \mathbb{R}^{n\times d_{out}}, \quad (1)$$

Self-attention as kernel smoothing

$$\mathrm{Attn}(x_v) = \sum_{u\in V} \frac{\kappa_{\exp}(x_v, x_u)}{\sum_{w\in V}\kappa_{\exp}(x_v, x_w)} f(x_u),\ \forall v \in V, \quad (3)$$

$$\kappa_{\exp}(x, x') := \exp\left(\langle \mathbf{W_Q}x, \mathbf{W_K}x'\rangle/\sqrt{d_{out}}\right), \quad (4)$$

Transformer

$$\mathbf{X}' = \mathbf{X} + \mathrm{Attn}(\mathbf{X}),$$
$$\mathbf{X}'' = \mathrm{FFN}(\mathbf{X}') := \mathrm{ReLU}(\mathbf{X}'W_1)W_2. \quad (2)$$

Structure-aware self-attention

$$\text{SA-attn}(v) := \sum_{u\in V} \frac{\kappa_{\text{graph}}(S_G(v), S_G(u))}{\sum_{w\in V}\kappa_{\text{graph}}(S_G(v), S_G(w))} f(x_u), \quad (5)$$

$$\kappa_{\text{graph}}(S_G(v), S_G(u)) = \kappa_{\exp}(\varphi(v, G), \varphi(u, G)), \quad (6)$$

k-subtree GNN extractor

$$\varphi(u, G) = \mathrm{GNN}_G^{(k)}(u). \quad (7)$$

k-subgraph GNN extractor

$$\varphi(u, G) = \sum_{v\in\mathcal{N}_k(u)} \mathrm{GNN}_G^{(k)}(v). \quad (8)$$

Structure-aware Transformer

$$x'_v = x_v + 1/\sqrt{d_v}\,\text{SA-attn}(v), \quad (9)$$

Chen, Dexiong, Leslie O'Bray, and Karsten Borgwardt. "**Structure-Aware Transformer for Graph Representation Learning**." In *ICML*, 2022.

GPS, NeurIPS, 2022

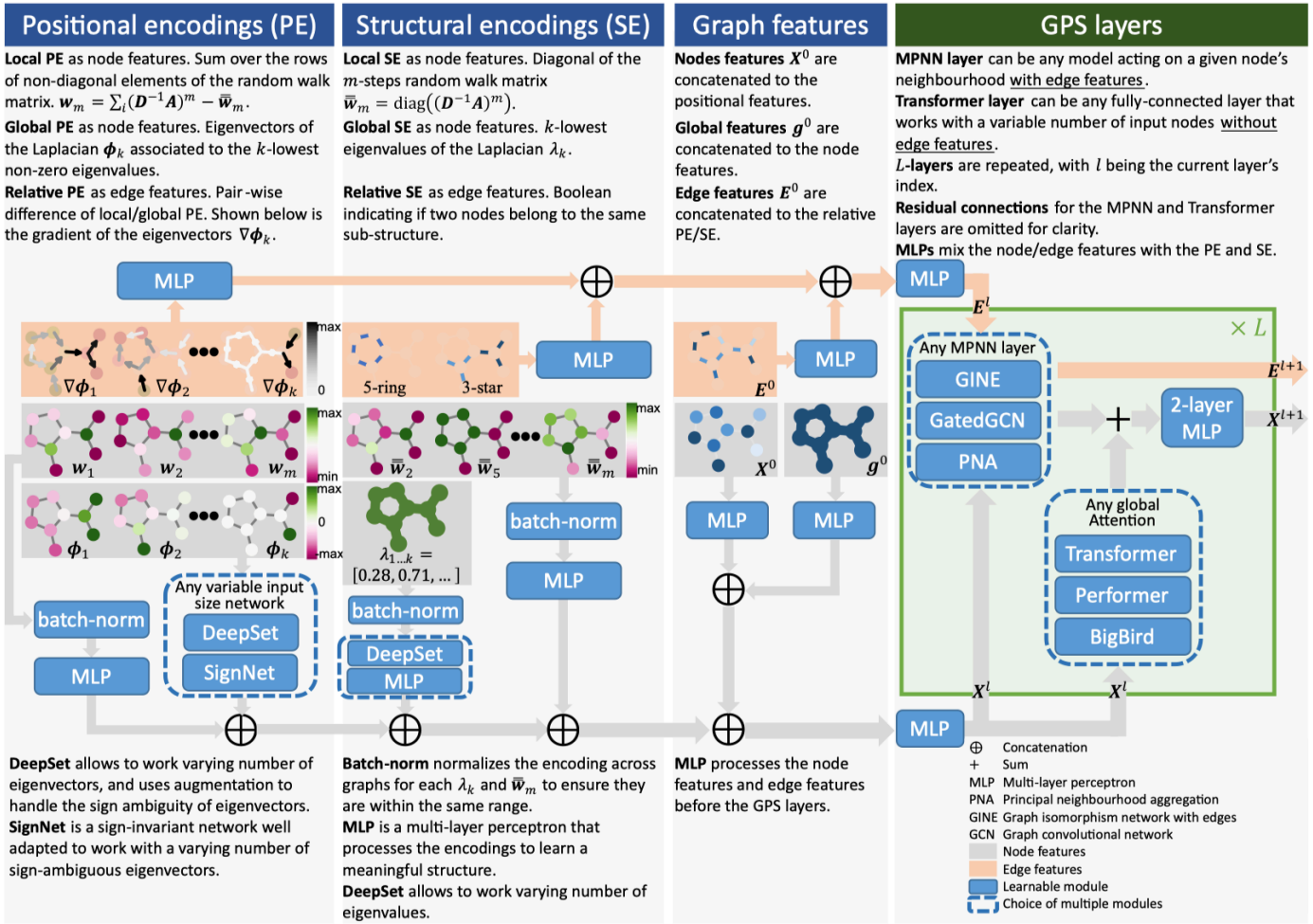


Figure 1: Modular GPS graph Transformer, with examples of PE and SE. Task specific layers for node/graph/edge-level predictions, such as pooling or output MLP, are omitted for simplicity.

GPS recipe consists of choosing 3 main ingredients:
(i) positional/structural encoding,
(ii) local message-passing mechanism
(iii) global attention mechanism

Table 1: The proposed categorization of positional encodings (PE) and structural encodings (SE). Some encodings are assigned to multiple categories in order to show their multiple expected roles.

Encoding type	Description	Examples
Local PE *node features*	Allow a node to know its position and role within a local cluster of nodes. *Within a **cluster**, the closer two nodes are to each other, the closer their local PE will be, such as the position of a word in a sentence (not in the text).*	• Sum each column of non-diagonal elements of the m-steps random walk matrix. • Distance between a node and the centroid of a cluster containing the node.
Global PE *node features*	Allow a node to know its global position within the graph. *Within a **graph**, the closer two nodes are, the closer their global PE will be, such as the position of a word in a text.*	• Eigenvectors of the Adjacency, Laplacian [15, 35] or distance matrices. • SignNet [38] (includes aspects of relative PE and local SE). • Distance from the graph's centroid. • Unique identifier for each connected component of the graph.
Relative PE *edge features*	Allow two nodes to understand their distances or directional relationships. *Edge embedding that is correlated to the distance given by any global or local PE, such as the distance between two words.*	• Pair-wise node distances [37, 3, 35, 62, 43] based on shortest-paths, heat kernels, random-walks, Green's function, graph geodesic, or any local/global PE. • Gradient of eigenvectors [3, 35] or any local/global PE. • PEG layer [56] (includes aspects of global PE). • Boolean indicating if two nodes are in the same cluster.
Local SE *node features*	Allow a node to understand what substructures it is a part of. *Given an SE of radius m, the more similar the m-hop subgraphs around two nodes are, the closer their local SE will be.*	• Degree of a node [62]. • Diagonal of the m-steps random-walk matrix [16]. • Time-derivative of the heat-kernel diagonal (gives the degree at $t = 0$). • Enumerate or count predefined structures such as triangles, rings, etc. [6, 67]. • Ricci curvature [53].
Global SE *graph features*	Provide the network with information about the global structure of the graph. *The more similar two graphs are, the closer their global SE will be.*	• Eigenvalues of the Adjacency or Laplacian matrices [35]. • Graph properties: diameter, girth, number of connected components, # of nodes, # of edges, nodes-to-edges ratio.
Relative SE *edge features*	Allow two nodes to understand how much their structures differ. *Edge embedding that is correlated to the difference between any local SE.*	• Pair-wise distance, encoding, or gradient of any local SE. • Boolean indicating if two nodes are in the same substructure [5] (similar to the gradient of sub-structure enumeration).

[1] Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, Dominique Beaini. "**Recipe for a General, Powerful, Scalable Graph Transformer**." In *NeurIPS*, 2022

Scalability of Graph Transformer

1. Node Sampling
2. Hierarchical
3. Pure Transformer

TokenGT, NeurIPS, 2022

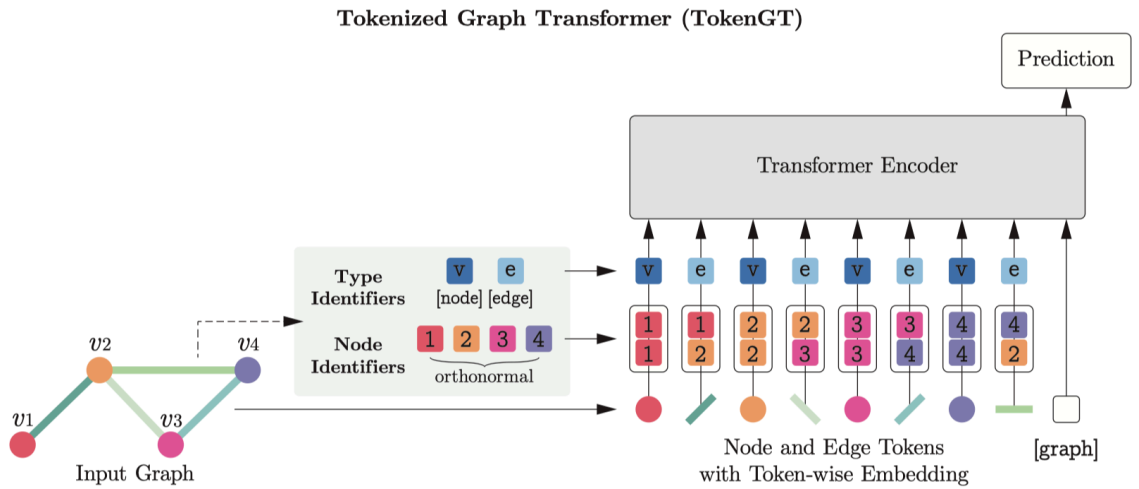


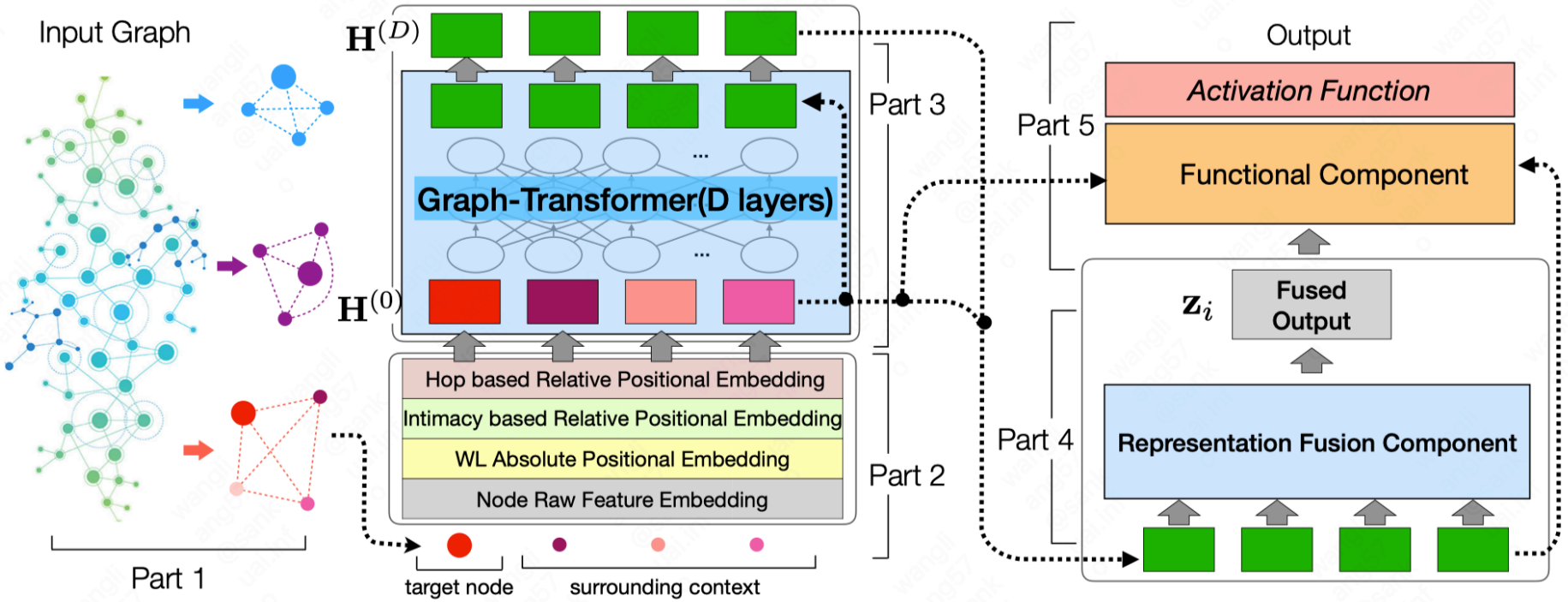
Figure 1: Overview of Tokenized Graph Transformer (TokenGT). We treat all nodes and edges of an input graph as independent tokens, augment them with orthonormal node identifiers and trainable type identifiers, and feed them to a standard Transformer encoder. For graph-level prediction, we follow the common practice [17, 18] of using an extra trainable `[graph]` token.

- For each node $v \in \mathcal{V}$, we augment the token $\mathbf{X}_v$ as $[\mathbf{X}_v, \mathbf{P}_v, \mathbf{P}_v]$.
- For each edge $(u, v) \in \mathcal{E}$, we augment the token $\mathbf{X}_{(u,v)}$ as $[\mathbf{X}_{(u,v)}, \mathbf{P}_u, \mathbf{P}_v]$.

Intuitively, a Transformer operating on the augmented tokens can fully recognize the connectivity structure of the graph since comparing the node identifiers between a pair of tokens reveals their *incidence* information. For instance, we can tell if an edge $e = (u, v)$ is connected with a node k through dot-product (attention) since $[\mathbf{P}_u, \mathbf{P}_v][\mathbf{P}_k, \mathbf{P}_k]^\top = 1$ if and only if $k \in (u, v)$ and 0 otherwise. This allows the Transformer to identify and exploit the connectivity structure of a graph, for instance by putting more weights on incident pairs when the local operation is important.

[1] Jinwoo Kim, Tien Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, Seunghoon Hong. "**Pure Transformers Are Powerful Graph Learners**." In *NeurIPS*, 2022

Graph-Bert, arXiv, 2020



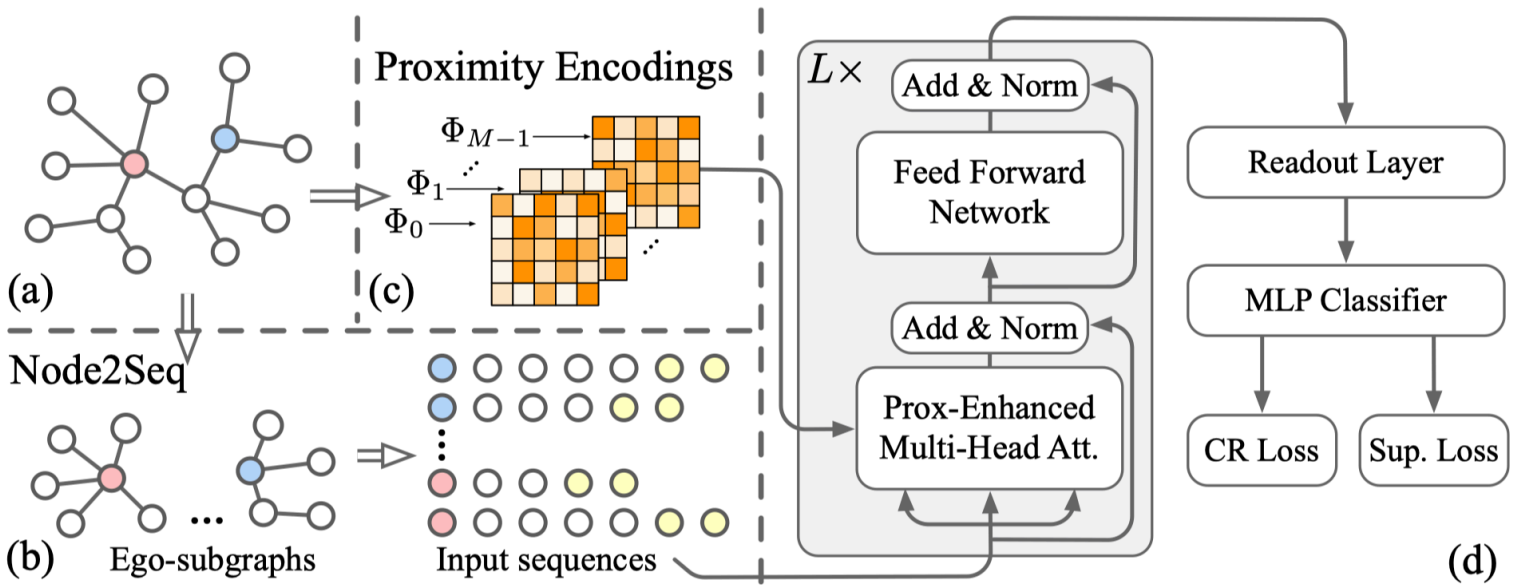
Linkless Subgraph Batching

$$\mathbf{S} = \alpha \cdot \left(\mathbf{I} - (1-\alpha) \cdot \bar{\mathbf{A}}\right)^{-1}, \quad (1)$$

$$\bar{\mathbf{A}} = \mathbf{A}\mathbf{D}^{-1}$$

[1] Jiawei Zhang, Haopeng Zhang, Congying Xia, Li Sun. "**Graph-Bert: Only Attention Is Needed for Learning Graph Representations**." arXiv. January 22, 2020

Gophormer, arXiv, 2021



$$\alpha_{ij} = \frac{\left(\boldsymbol{h}_i \mathbf{W}_Q\right)\left(\boldsymbol{h}_j \mathbf{W}_K\right)^T}{\sqrt{d}} + \boldsymbol{\phi}_{ij}^T \boldsymbol{b}, \tag{5}$$

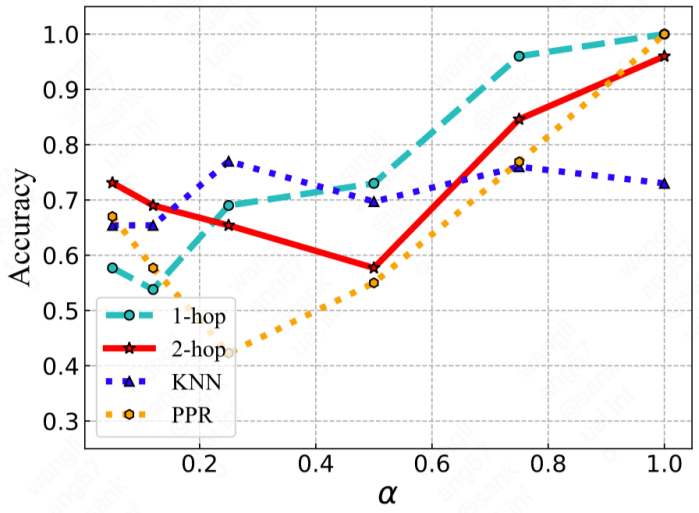
$$\boldsymbol{\phi}_{ij} = \text{Concat}(\Phi_m(v_i, v_j) | m \in 0, 1, .., M-1), \tag{6}$$

$$\Phi_m(v_i, v_j) = \begin{cases} \tilde{\mathbf{A}}^m[i,j], & \text{if } m < M-1 \\ \mathbb{I}(i,j), & \text{if } m = M-1 \end{cases}, \tag{7}$$

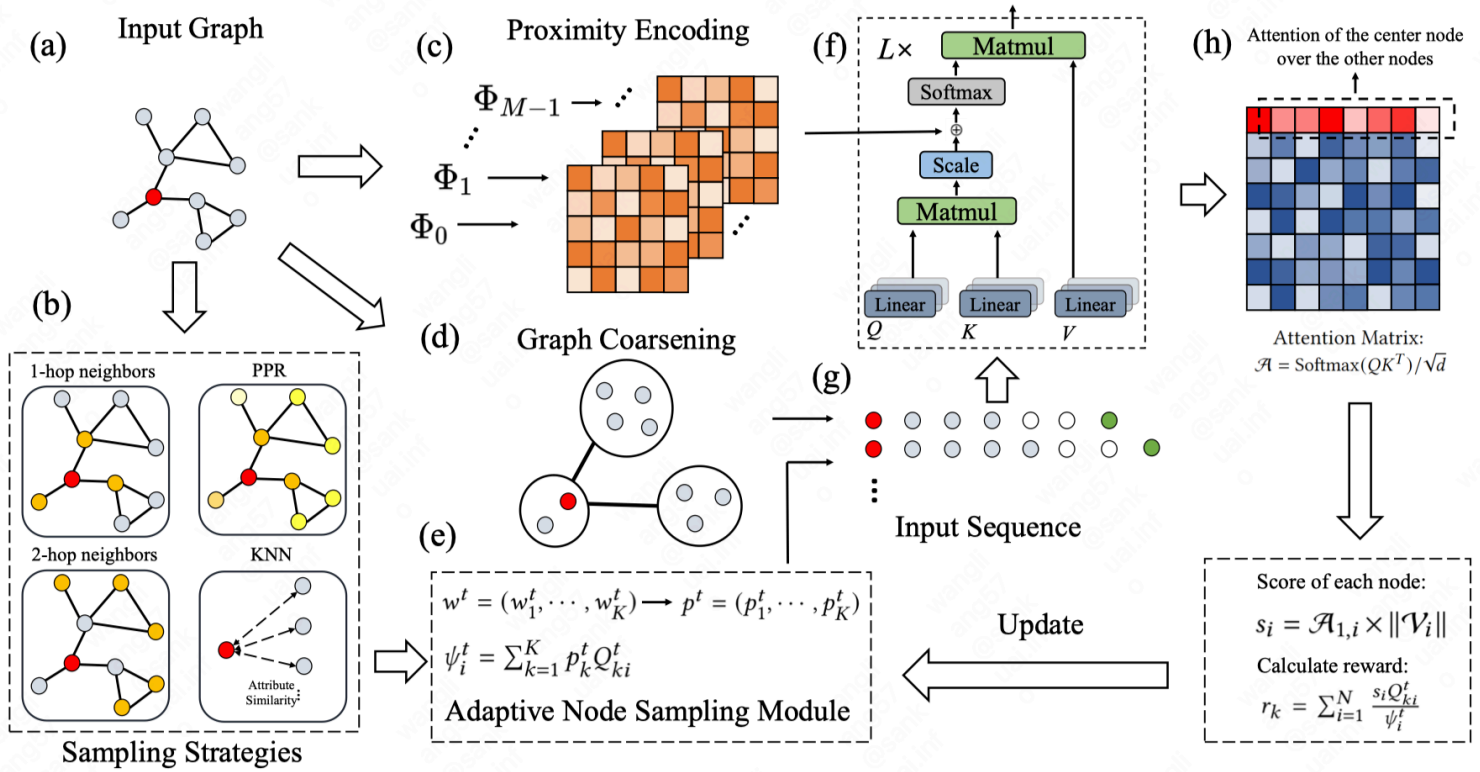
Figure 1: Model framework of Gophormer. (a) A sample graph with two example nodes colored red and blue. (b) The Node2Seq process: ego-graphs are sampled from the original graph and converted to sequential data. White nodes are context nodes, yellow nodes are global nodes to store graph-level context. In the example two sample ego-graphs are drawn for each node. (c) The proximity encoding process. (d) The main graph transformer framework of Gophormer.

[1] Jianan Zhao, Chaozhuo Li, Qianlong Wen, Yiqi Wang, Yuming Liu, Hao Sun, Xing Xie, Yanfang Ye. "**Gophormer: Ego-Graph Transformer for Node Classification**." *ArXiv*, 2021

ANS-GT, NeurIPS, 2022



Degree of homophily: $\alpha \triangleq \frac{|(v_i,v_j)\in E \wedge y_i = y_j|}{|E|}$



Deficiencies of existing node sampling strategies:

(i) Ignorant of the graph properties → (i) Adaptive Node Sampling (formulate the optimization strategy of node sampling as an adversary bandit problem)

(ii) Focus on local neighbors and neglect the long-range dependencies and global contexts → (ii) Graph Coarsening

[1] Zaixi Zhang, Qi Liu, Qingyong Hu, Chee-Kong Lee. "**Hierarchical Graph Transformer with Adaptive Node Sampling**." In *NeurIPS*. 2022

Higher-Order Transformer, NeurIPS, 2021

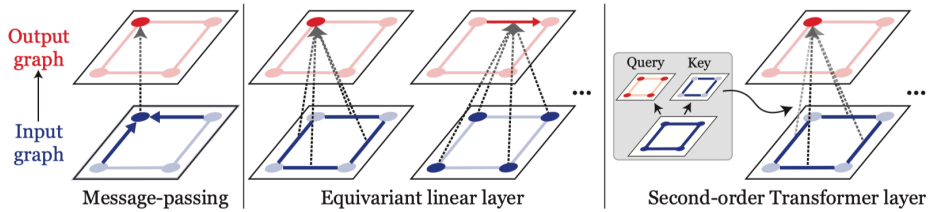


Figure 1: Illustrated operations of a message-passing GNN, an equivariant linear layer, and a second-order Transformer layer. A single output node is highlighted.

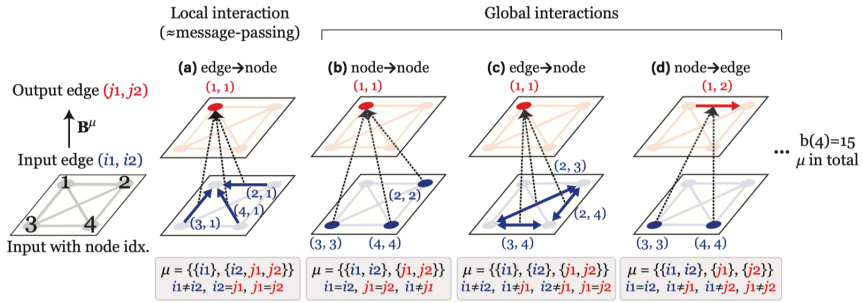


Figure 2: Example operations in a second-order layer $L_{2\to 2}$. We illustrate how input edges $(i_1, i_2) = \mathbf{i}$ are aggregated to an output edge $(j_1, j_2) = \mathbf{j}$ in various equivalence classes $(\mathbf{i}, \mathbf{j}) \in \mu$ via basis tensor $\mathbf{B}^{\mu}_{\mathbf{i},\mathbf{j}}$. Note that a loop $((i_1, i_1)$ or $(j_1, j_1))$ represents a node.

Invariant and equivariant linear layer

$$L_{k\to l}(\mathbf{A})_{\mathbf{j}} = \sum_{\mu}\sum_{\mathbf{i}} \mathbf{B}^{\mu}_{\mathbf{i},\mathbf{j}}\mathbf{A}_{\mathbf{i}} w_{\mu} + \sum_{\lambda} \mathbf{C}^{\lambda}_{\mathbf{j}} b_{\lambda}, \tag{1}$$

- Invariant and Equivariant Graph Networks. In *ICLR*, 2019
- Expressive Power of Invariant and Equivariant Graph Neural Networks. In *ICLR*, 2021

Eq. 9 is a generalization of Eq. 1

HOT layer

and adding feedforward MLP. In this section, we generalize the approach to $L_{k\to l}$ with arbitrary orders k and l and formulate higher-order Transformer layer $\text{Enc}_{k\to l}$.

In general, we define higher-order Transformer layer as a function $\text{Enc}_{k\to l} : \mathbb{R}^{n^k \times d} \to \mathbb{R}^{n^l \times d}$ with two layers: a higher-order self-attention layer $\text{Attn}_{k\to l} : \mathbb{R}^{n^k \times d} \to \mathbb{R}^{n^l \times d}$ and a feedforward layer $\text{MLP}_{l\to l} : \mathbb{R}^{n^l \times d} \to \mathbb{R}^{n^l \times d}$. For an input tensor $\mathbf{A} \in \mathbb{R}^{n^k \times d}$, a Transformer layer computes:

$$\text{MLP}_{l\to l}(\text{Attn}_{k\to l}(\mathbf{A})) = L^2_{l\to l}\left(\text{ReLU}(L^1_{l\to l}(\text{Attn}_{k\to l}(\mathbf{A})))\right), \tag{8}$$

$$\text{Enc}_{k\to l}(\mathbf{A}) = \text{Attn}_{k\to l}(\mathbf{A}) + \text{MLP}_{l\to l}(\text{Attn}_{k\to l}(\mathbf{A})), \tag{9}$$

where $L^1_{l\to l} : \mathbb{R}^{n^l \times d \to n^l \times d_F}$ and $L^2_{l\to l} : \mathbb{R}^{n^l \times d_F \to n^l \times d}$ are equivariant linear layers with hidden dimension d_F. Remaining question is how to define and compute higher-order self-attention $\text{Attn}_{k\to l}(\mathbf{A})$.

Attn

$$\text{Attn}_{k\to l}(\mathbf{A})_{\mathbf{j}} = \sum_{h=1}^{H}\sum_{\mu}\sum_{\mathbf{i}} \boldsymbol{\alpha}^{h,\mu}_{\mathbf{i},\mathbf{j}} \mathbf{A}_{\mathbf{i}} w^{V}_{h,\mu} w^{O}_{h,\mu}, \tag{10}$$

$$\boldsymbol{\alpha}^{\mu}_{\mathbf{i},\mathbf{j}} = \begin{cases} \sigma(\mathbf{Q}^{\mu}_{\mathbf{j}}, \mathbf{K}^{\mu}_{\mathbf{i}})/Z_{\mathbf{j}} & (\mathbf{i},\mathbf{j}) \in \mu \\ 0 & \text{otherwise} \end{cases} \quad \text{where } \mathbf{Q}^{\mu} = L^{\mu}_{k\to l}(\mathbf{A}), \mathbf{K}^{\mu} = L^{\mu}_{k\to k}(\mathbf{A}). \tag{12}$$

[1] Jinwoo Kim, Saeyoon Oh, Seunghoon Hong. "**Transformers Generalize DeepSets and Can Be Extended to Graphs & Hypergraphs**." In *NeurIPS*. 2021

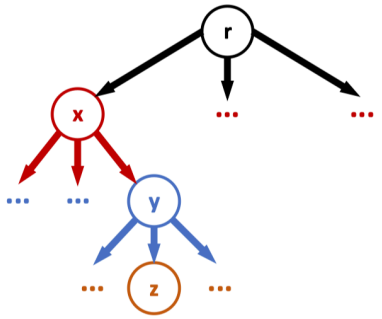
Tree-Transformer, NeurIPS, 2019

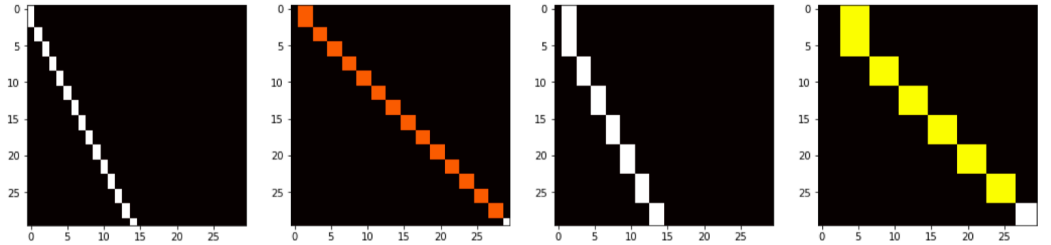
Aims:

1. uniqueness property
2. positional relationship property

Figure 1: Example computations of positional encodings for nodes in a regular tree. The sequence of branch choices b determines a sequence of transforms $D_{b_1}, D_{b_2}, \ldots$ to apply to the root node's positional encoding. U is defined as the complement: applying it to any node results in that node's parent (e.g. $r = Ux = U^2y = U^3z$). The transforms D_i, U are defined in Equations 2 and 3.

$$
\begin{aligned}
r &= \mathbf{0} &&= [0,0,0,0,0,0,0,0,0,0] \\
x &= D_0\mathbf{0} &&= [1,0,0,0,0,0,0,0,0,0] \\
y &= D_2D_0\mathbf{0} &&= [0,0,1,1,0,0,0,0,0,0] \\
z &= D_1D_2D_0\mathbf{0} &&= [0,1,0,0,0,1,1,0,0,0]
\end{aligned}
$$





(a) Parent: P (b) Siblings: $\frac{D_1U+D_2U}{2}$ (c) Aunts: $\frac{(D_1+D_2)U^2}{2}$ (d) Cousins: $\frac{(D_1+D_2)^2U^2}{4}$

choice at the ith layer and L is the layer at which the node resides. Then, for any node x, we compute its positional encoding as demonstrated in Figure 1:

$$x = D_{b_L} D_{b_{L-1}} \ldots D_{b_1} \mathbf{0}$$

complementarily. In other words, for a given node x,

$$D_i x = e_i^n ; x[:-n] \quad (2)$$

$$Ux = x[n:]; \mathbf{0}_n \quad (3)$$

[1] Vighnesh Shiv, Chris Quirk. "**Novel Positional Encodings to Enable Tree-Based Transformers.**" In *NeurIPS*. 2019

GNN + Transformer in NLP

GraphFormer, arXiv, 2021

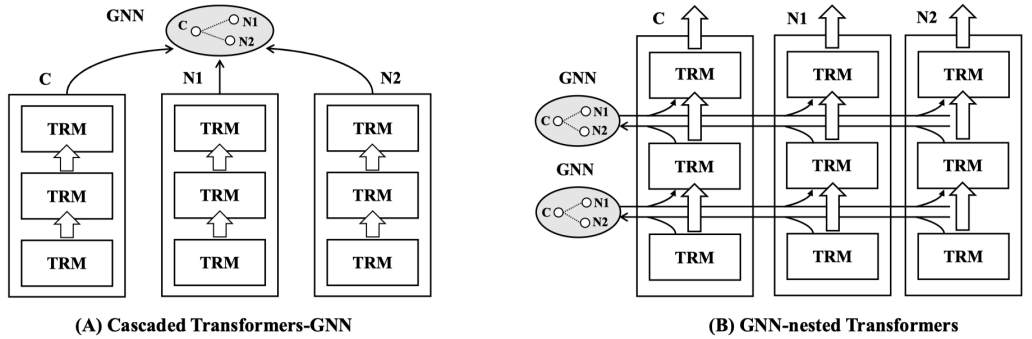


Figure 1: Model architecture comparison (a center node C is connected with two neighbours N1, N2). (A) Cascaded Transformers-GNN: text embeddings are independently generated by language models and aggregated by rear-mounted GNNs. (B) GNN-nested Transformers: the text encoding and graph aggregation are iteratively performed with the layerwise GNNs and Transformers (TRM).

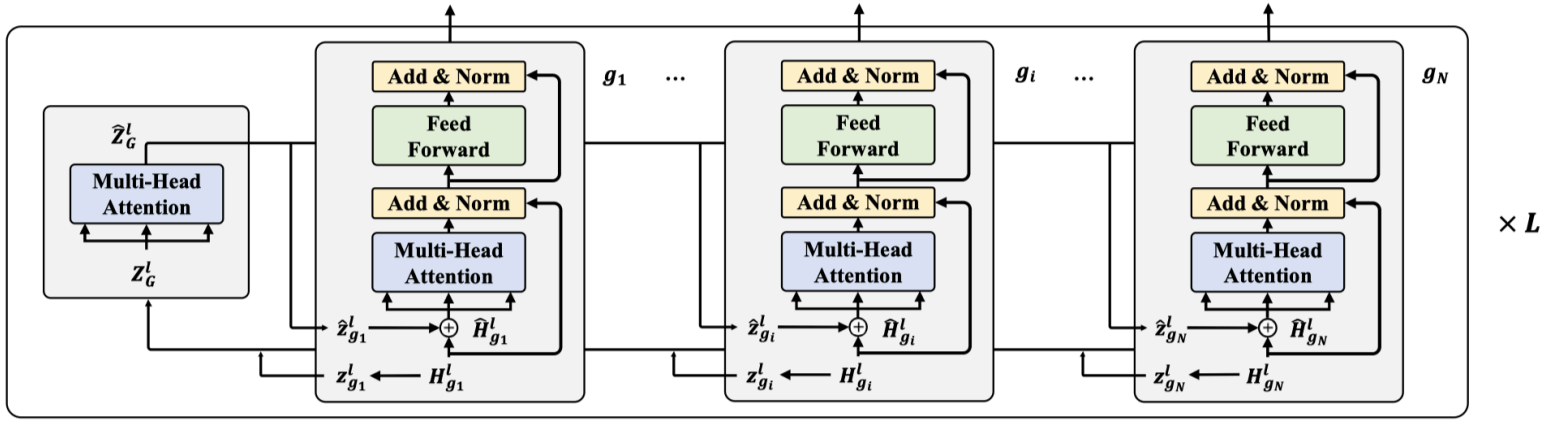


Figure 2: GNN-nested Transformers (using the l-th layer for illustration). The graph aggregation is performed in the first place: the node-level embeddings $\{\mathbf{z}_g^l\}_G$ are gathered from all the nodes and processed by the GNN component (the leftmost rectangle). The GNN processed node-level embeddings $\{\hat{\mathbf{z}}_g^l\}_G$ are dispatched to their original nodes, which forms the graph-augmented token-level embeddings. The graph-augmented token-level embeddings are further encoded by Transformer.

[1] Junhan Yang, Zheng Liu, Shitao Xiao, Chaozhuo Li, Defu Lian, Sanjay Agrawal, Amit Singh, Guangzhong Sun, Xing Xie. "**GraphFormers: GNN-Nested Transformers for Representation Learning on Textual Graph**." arXiv. September 30, 2021

Heterformer, arXiv, 2022

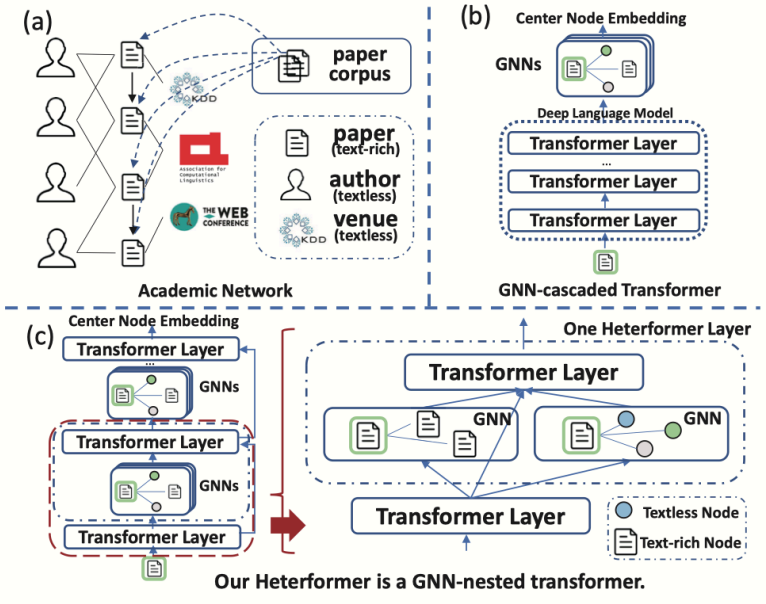


Figure 1: (a) An example of heterogeneous text-rich networks: an academic network. (b) An illustration of GNN-cascaded transformer. (c) An illustration of our GNN-nested transformer, Heterformer. One Heterformer layer is zoomed out.

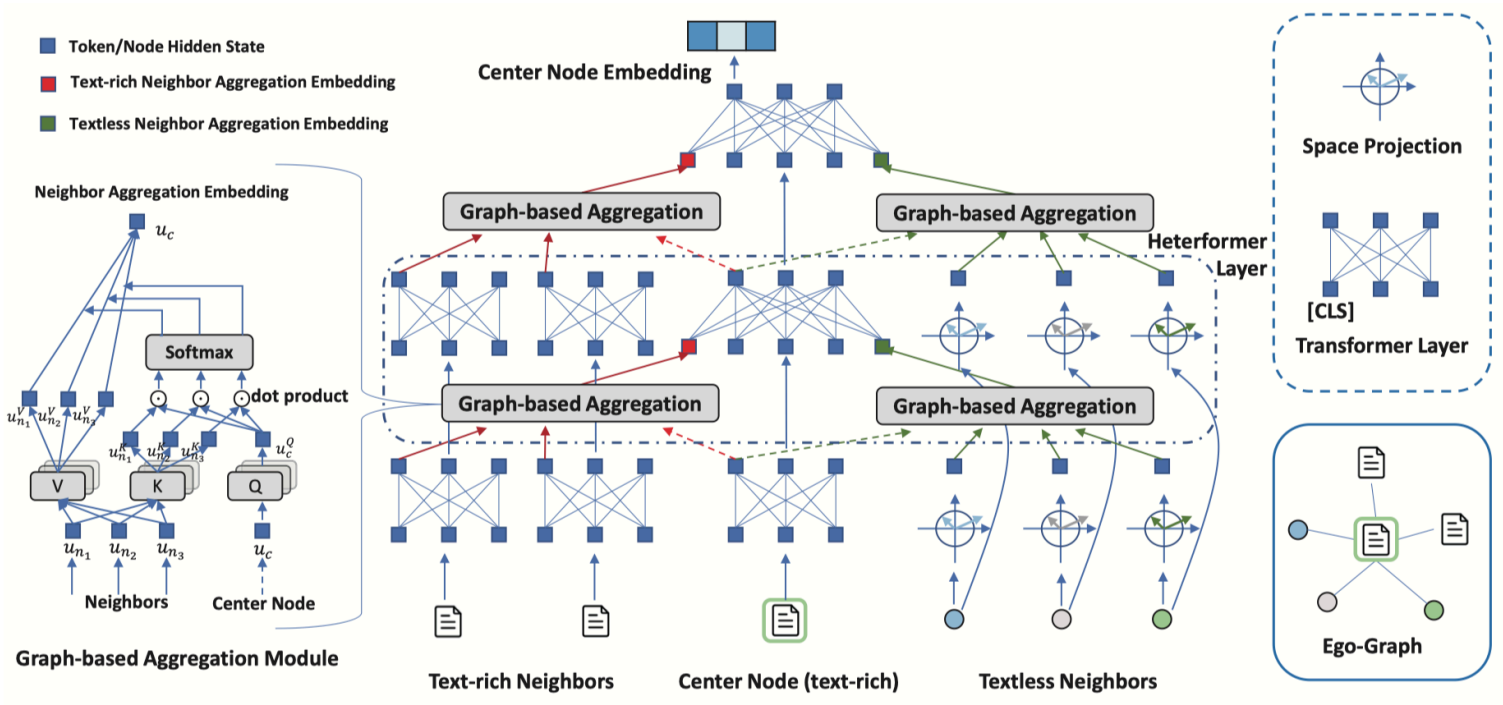


Figure 2: The overall architecture of Heterformer. There are two layers in the figure, while in experiments we have 11 layers.

[1] Bowen Jin, Yu Zhang, Qi Zhu, Jiawei Han. "**Heterformer: A Transformer Architecture for Node Representation Learning on Heterogeneous Text-Rich Networks**." arXiv. May 20, 2022

Application of Graph Transformer

Matformer, NeurIPS, 2022

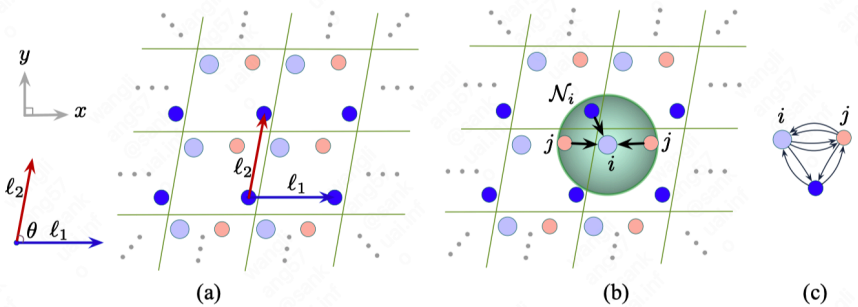


Figure 1: Illustrations of periodic patterns and the multi-edge graph construction method. Green

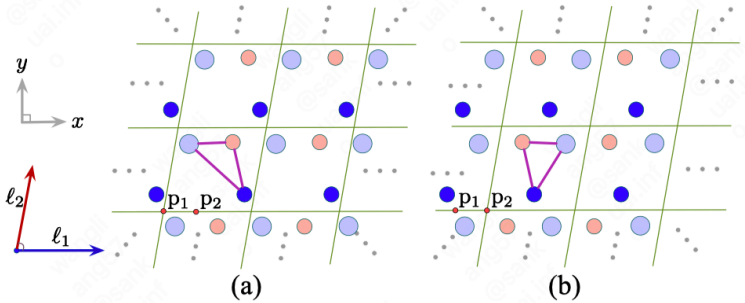


Figure 2: Illustration of periodic invariance. Purple lines are

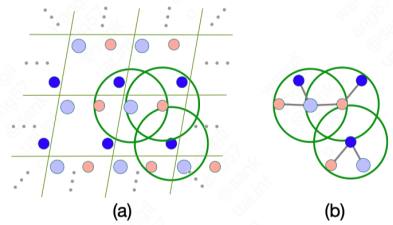


Figure 3: Illustration that periodic patterns are not encoded in a multi-edge graph. Grey lines show the

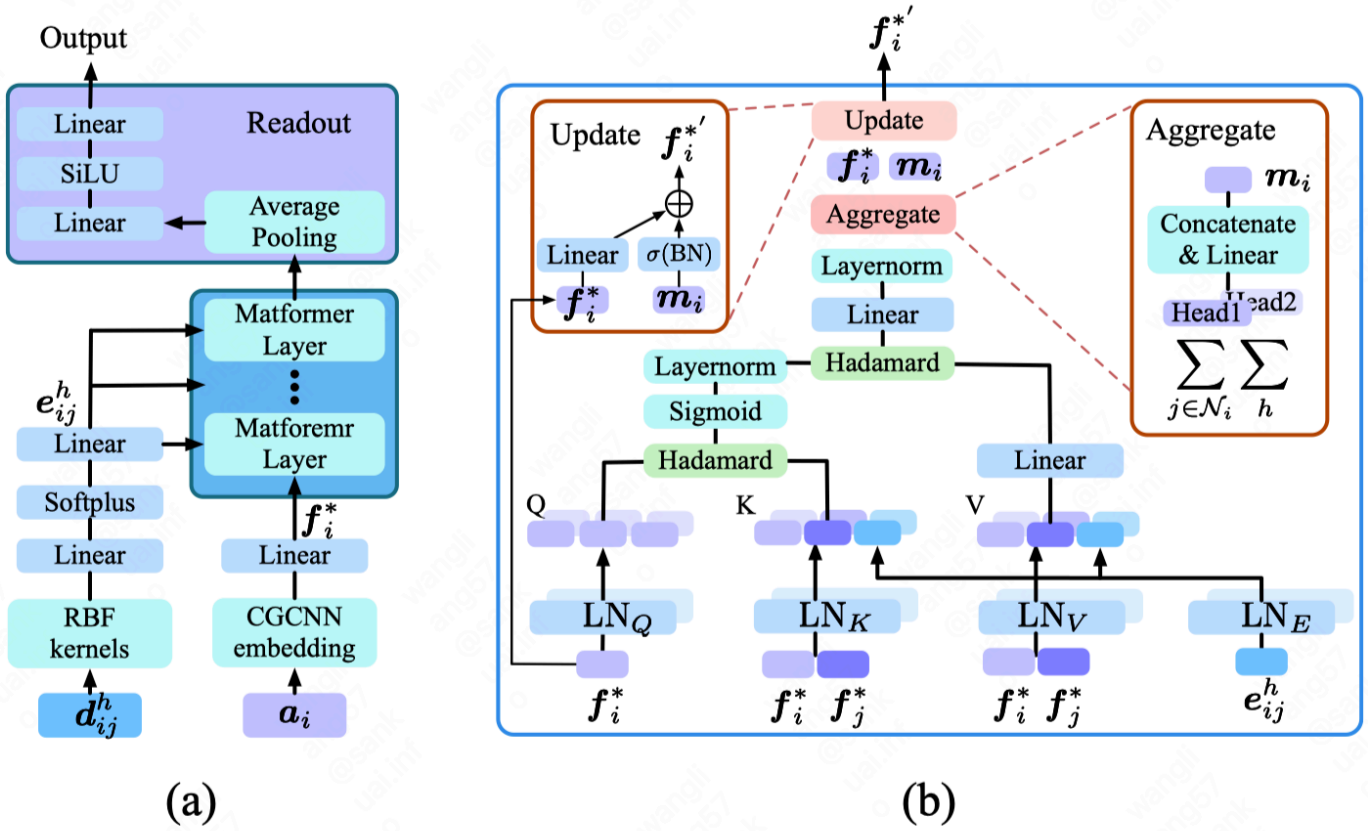
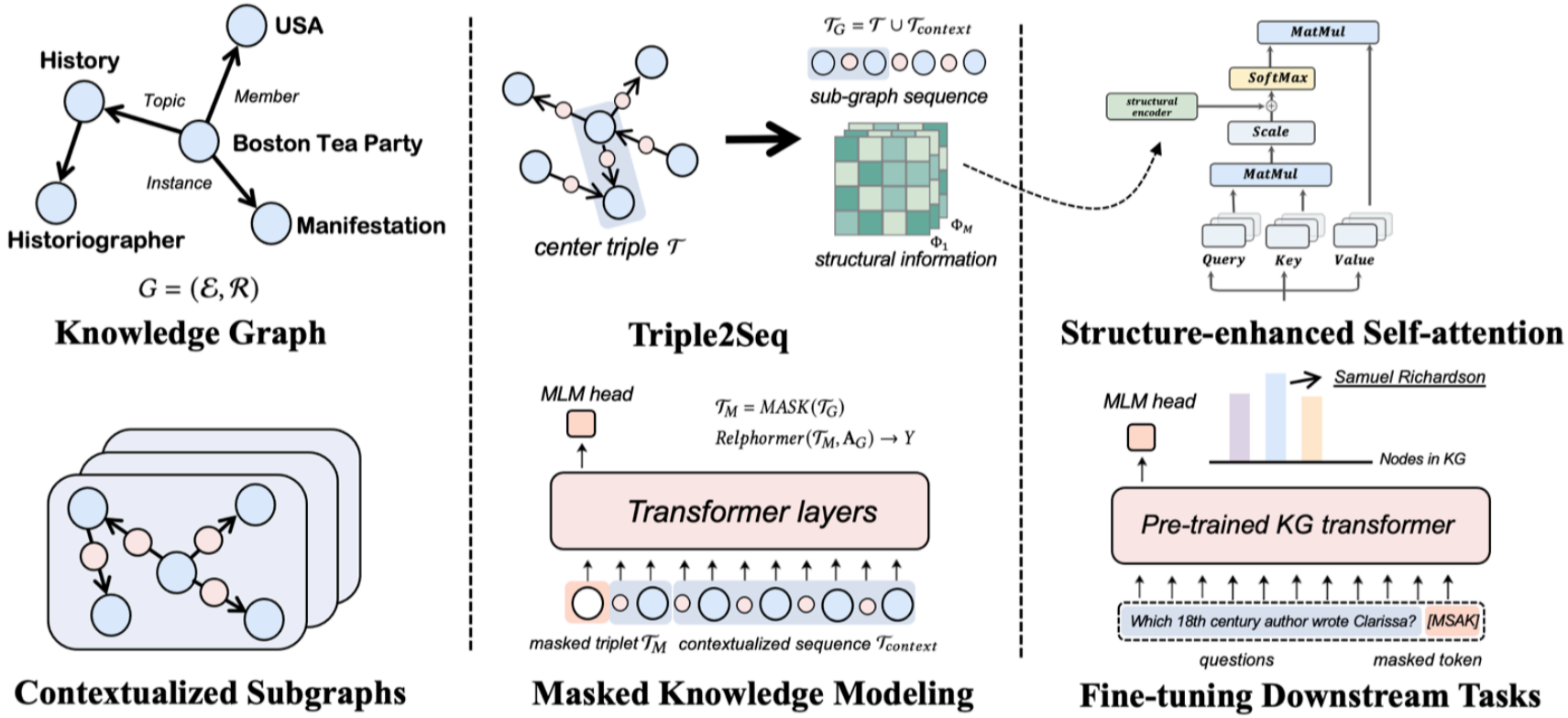


Figure 5: Illustration of detailed Architecture of Matformer. The overlapping graphics are used to denote two different attention heads. (a). Illustration of Matformer pipeline. (b). Illustration of the detailed Matformer layer in (a). We show the case with two attention heads for simplicity.

[1] Keqiang Yan, Yi Liu, Yuchao Lin, Shuiwang Ji. "**Periodic Graph Transformers for Crystal Material Property Prediction**." In *NeurIPS*. 2022

Relphormer, arXiv, 2022



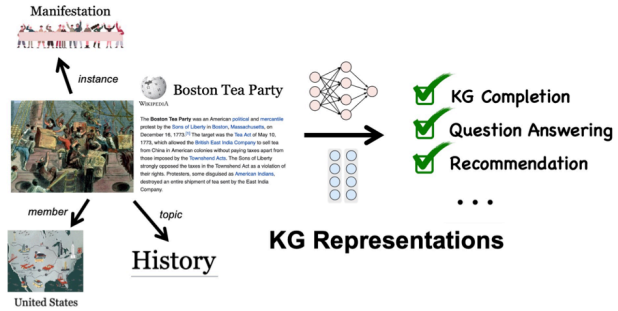


Figure 1: Examples from WN18RR and Wikipedia. KG representations can be applied to various tasks.

Challenges of applying Transformer to KGs:

(i) Heterogeneity for edges and nodes → (i) Triple2Seq, Dynamic sampling strategy, Structure-enhanced self-attention mechanism

(ii) Task Optimization Universality → (ii) Mask knowledge modeling mechanism

[1] Zhen Bi, Siyuan Cheng, Jing Chen, Xiaozhuan Liang, Ningyu Zhang, Feiyu Xiong, Huajun Chen. "Relphormer: Relational Graph Transformer for Knowledge Graph Representations." arXiv. October 15, 2022

Quick Look at ICLR 2023 Submissions

Total: 13

- Scalability: 4
- Expressive Power: 2
- Positional Encoding: 2 (substructure, curvature)
- Relational Graph: 1
- Graph Structure Learning: 1
- Spectral Domain: 1
- NAS: 1
- Energy-based Model: 1

~~Exphormer, ICLR 2023 Submission~~

$O(|E|)$ $O(|V|)$ $O(|V|)$ $O(|V| + |E|)$

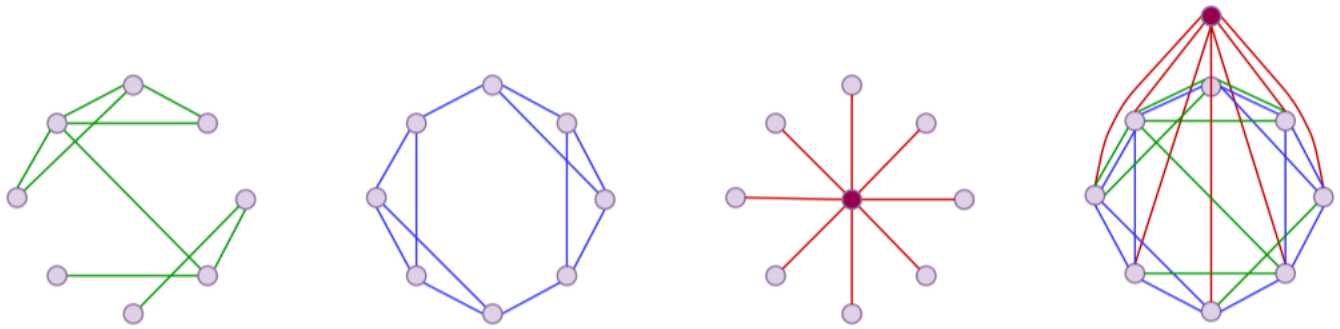


Figure 1: The components of EXPHORMER: (a) shows local neighborhood attention, i.e., edges of the input graph. (b) shows an expander graph with degree 3. (c) shows global attention with a single virtual node. (d) All of the aforementioned components are combined into a single interaction graph that determines the attention pattern of EXPHORMER.

•Sparse attention w/ expander graph mask, linear complexity with nice theoretical properties (spectrally similar to dense attention, few layers needed for all pairs of nodes to communicate).

[1] Anonymous. "**Exphormer: Scaling Graph Transformers with Expander Graphs.**" In *ICLR 2023 Submission*, 2022

~~GOAT, ICLR 2023 Submission~~

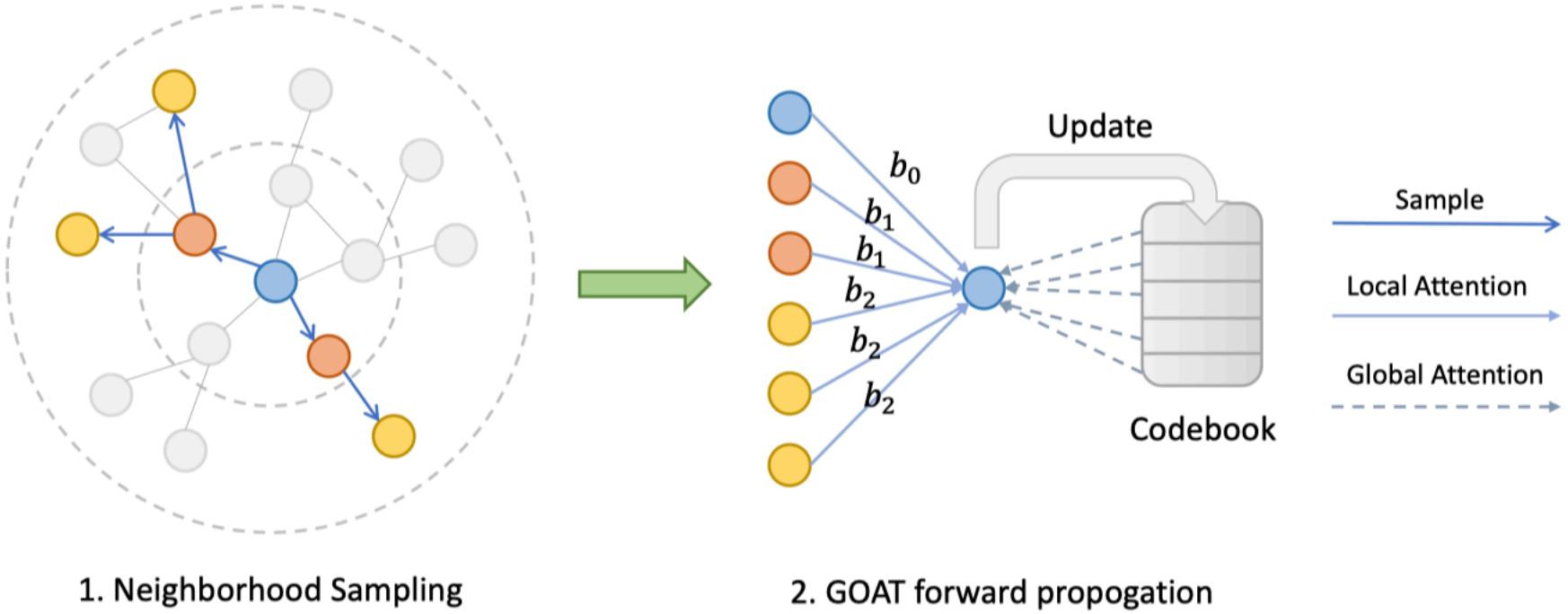


Figure 1: The local sampling procedure and forward propogation of the GOAT model. b_k denotes the trainable positional bias for neighbors at a distance of k.

•Computes attention between nodes and a **fixed size** codebook, and local attention on sampled k-hop neighborhoods, works on homophilous / non-homophilous graphs w/ > 1 millions nodes.

[1] Anonymous. "**GOAT: A Global Transformer on Large-Scale Graphs.**" In *ICLR 2023 Submission*, 2022

NAGphormer, ICLR 2023 Submission

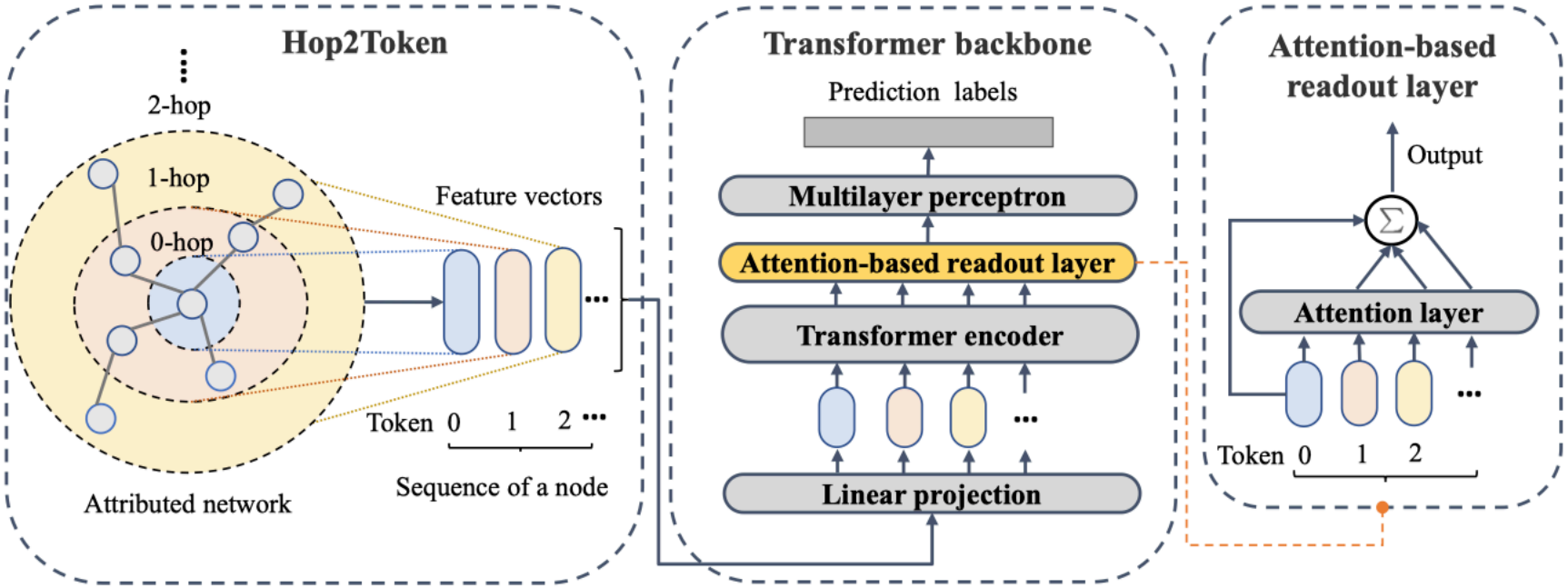


Figure 1: **Model framework of NAGphormer.** NAGphormer first uses a novel neighborhood aggregation module, Hop2Token, to construct a sequence for each node based on the tokens of different hops of neighbors. Then, NAGphormer learns the node representations using a Transformer backbone, and an attention-based readout function is developed to aggregate neighborhood information of different hops adaptively. An MLP-based module is used in the end for label prediction.

[1] Anonymous. "**NAGphormer: A Tokenized Graph Transformer for Node Classification in Large Graphs**." In *ICLR 2023 Submission*, 2022

~~DET, ICLR 2023 Submission~~

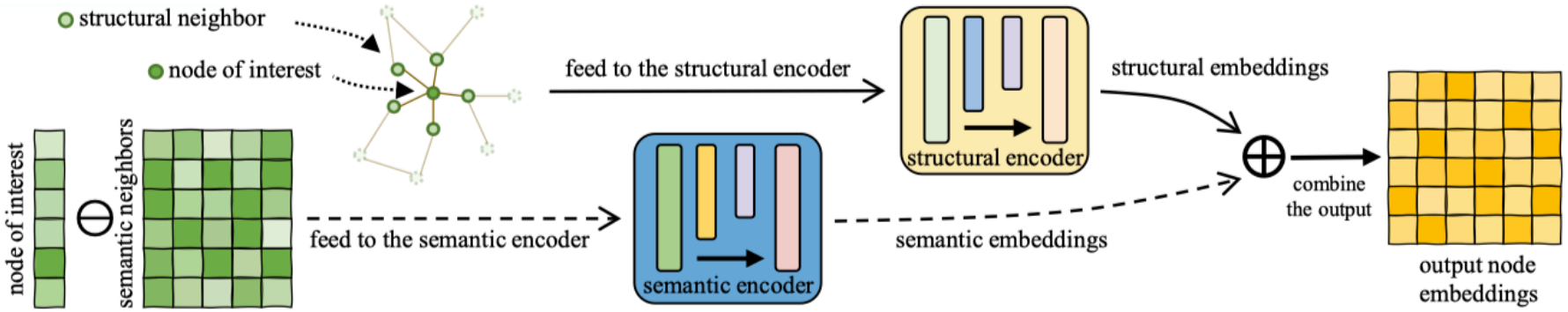
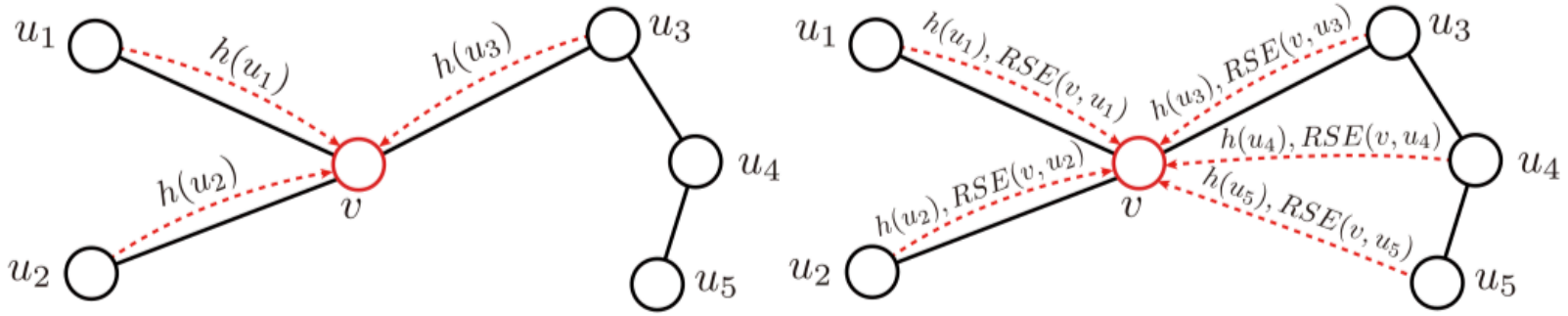


Figure 1: Overview of DET. Structural neighbors are local neighbors connected with the node of interest on the graph, while semantic neighbors are remote nodes with similar semantics to the node of interest. The two encoders focus on encoding different aspects of neighboring information, and thus are capable of complementing each other.

•Local attention on 1-hop neighbors (with PEs) + global attention with most semantically similar nodes (in terms of feature similarity). Does node / edge / graph tasks.

[1] Anonymous. "**Reach the Remote Neighbors: Dual-Encoding Transformer for Graphs.**" In *ICLR 2023 Submission*, 2022

~~GT Test, ICLR 2023 Submission~~



(a) WL Test (b) GT Test

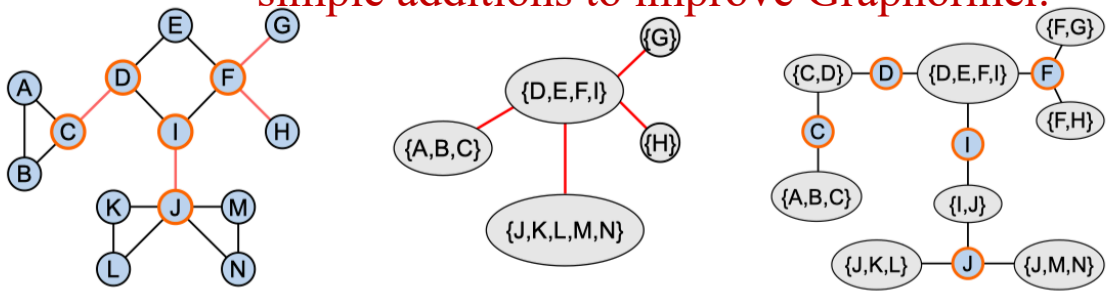
Figure 1: An illustration of the node label update strategies of WL test and GT test.

•Formulates a graph isomorphism test that bounds the power of Graph Transformers, with power determined by choice of absolute / relative PEs.
Proposes new shortest-path-related relative PE.

[1] Anonymous. "**On Structural Expressive Power of Graph Transformers**." In *ICLR 2023 Submission*, 2022

GD-WL, ICLR 2023 Submission

- New theoretical framework for analyzing GNNs, leads to simple additions to improve Graphormer.



(a) Original graph (b) Block cut-edge tree (c) Block cut-vertex tree

Figure 1: An illustration of edge-biconnectivity and vertex-biconnectivity. Cut vertices/edges are outlined in bold red. Gray nodes in (b)/(c) are edge/vertex-biconnected components, respectively.

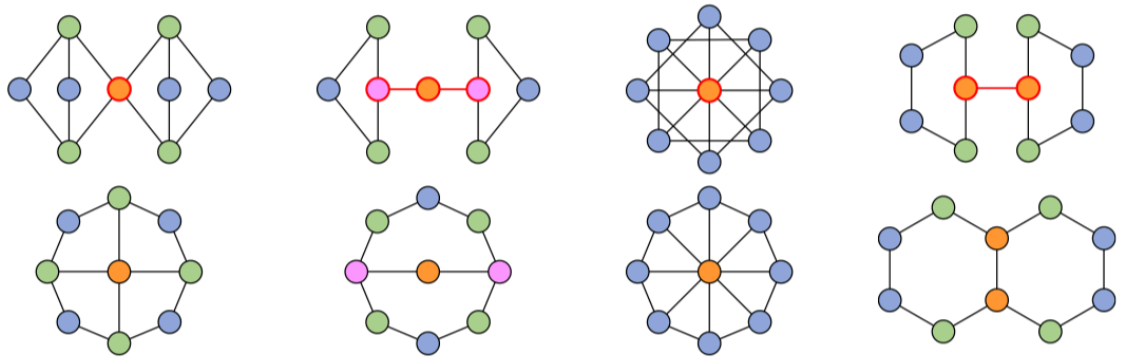


Figure 2: Illustration of four representative counterexamples (see Examples C.9 and C.10 for a ge eral definition). Graphs in the first row have cut vertices (outlined in bold red) and some also have c edges (denoted as red lines), while graphs in the second row do not have any cut vertex or cut edg

Model	Cut Vertex Detection	Cut Edge Detection
GCN (Kipf & Welling, 2017)	50.8%	61.9%
GAT (Veličković et al., 2018)	51.1%	62.2%
GIN (Xu et al., 2019)	52.3%	63.6%
GSN (Bouritsas et al., 2022)	58.9%	69.4%
Graphormer (Ying et al., 2021a)	75.4%	82.5%
Graphormer-GD (ours)	100%	100%
- w/o. Resistance Distance	84.4%	100%

Table 2: Mean Absolute Error (MAE) on ZINC test set. Following Dwivedi et al. (2020), the parameter budget of compared models is set to 500k. We use * to indicate the best performance.

Method	Model	Test MAE ZINC-Subset	Test MAE ZINC-Full
MPNNs	GIN (Xu et al., 2019)	0.526±0.051	0.088±0.002
	GraphSAGE (Hamilton et al., 2017)	0.398±0.002	0.126±0.003
	GAT (Veličković et al., 2018)	0.384±0.007	0.111±0.002
	GCN (Kipf & Welling, 2017)	0.367±0.011	0.113±0.002
	MoNet (Monti et al., 2017)	0.292±0.006	0.090±0.002
	GatedGCN-PE (Bresson & Laurent, 2017)	0.214±0.006	-
	MPNN(sum) (Gilmer et al., 2017)	0.145±0.007	-
	HIMP (Fey et al., 2020)	0.151±0.006	0.036±0.002
	PNA (Corso et al., 2020)	0.142±0.010	-
Substructure Count	GSN (Bouritsas et al., 2022)	0.101±0.010	-
Cellular Complex	CIN-Small (Bodnar et al., 2021a)	0.094±0.004	-
Subgraph GNNs	NGNN (Zhang & Li, 2021)	0.111±0.003	-
	DS-GNN (EGO) (Bevilacqua et al., 2022)	0.115±0.004	-
	DS-GNN (EGO+) (Bevilacqua et al., 2022)	0.105±0.003	-
	DSS-GNN (EGO) (Bevilacqua et al., 2022)	0.099±0.003	-
	DSS-GNN (EGO+) (Bevilacqua et al., 2022)	0.097±0.006	-
	GNN-AK (Zhao et al., 2022)	0.105±0.010	-
	GNN-AK-CTX (Zhao et al., 2022)	0.093±0.002	-
	GNN-AK+ (Zhao et al., 2022)	0.091±0.011	-
Graph Transformers	GT (Dwivedi & Bresson, 2021)	0.226±0.014	-
	SAN (Kreuzer et al., 2021)	0.139±0.006	-
	Graphormer (Ying et al., 2021a)	0.122±0.006	0.052±0.005
GD-WL	Graphormer-GD (ours)	0.081±0.009*	0.025±0.004*

[1] Anonymous. "**Rethinking the Expressive Power of GNNs via Graph Biconnectivity.**" In *ICLR 2023 Submission*, 2022

DeepGraph, ICLR 2023 Submission

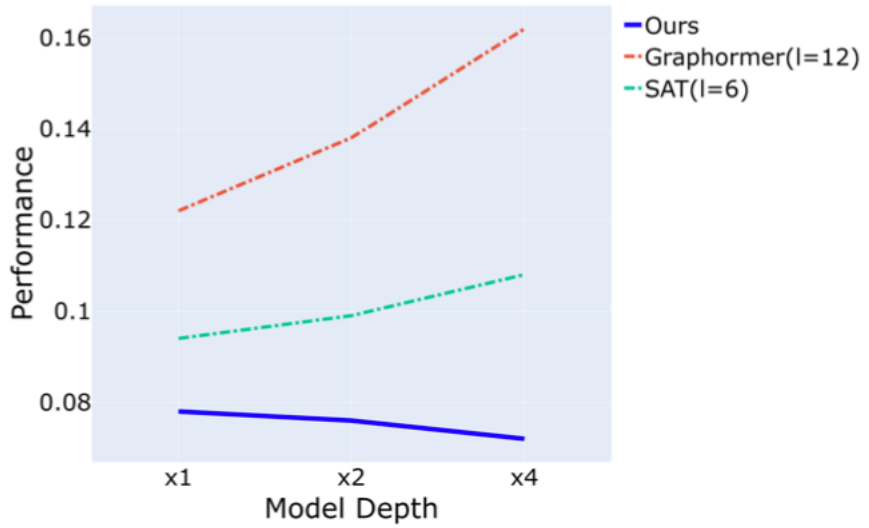


Figure 1: Performance on ZINC dataset of different graph transformers by varying their depths. (Lower is better.)

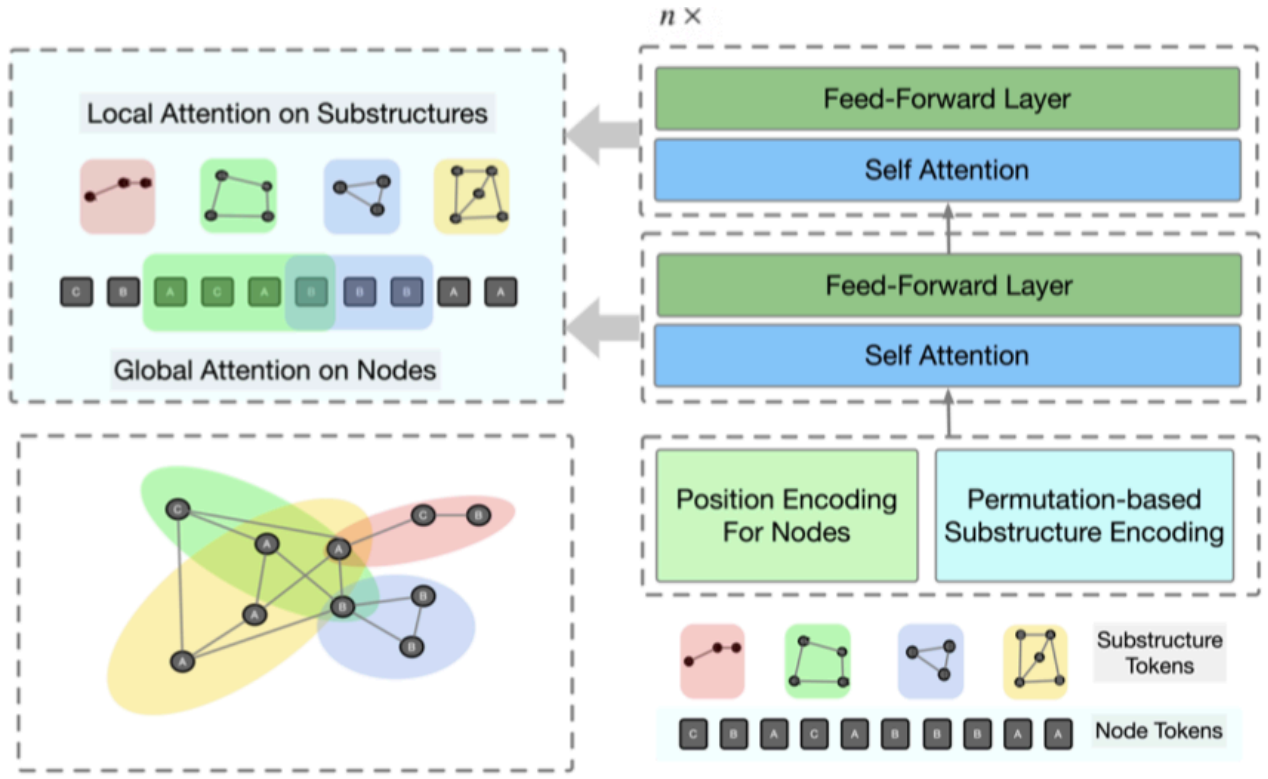


Figure 2: Overview of the proposed graph encoding framework.

•No. Existing Graph Transformers are often worse when too deep. Authors propose adding subgraph tokens with local attention, and find that depth then increases performance.

[1] Anonymous. "**Are More Layers Beneficial to Graph Transformers?**" In *ICLR 2023 Submission*, 2022

~~Curvphormer, ICLR 2023 Submission~~

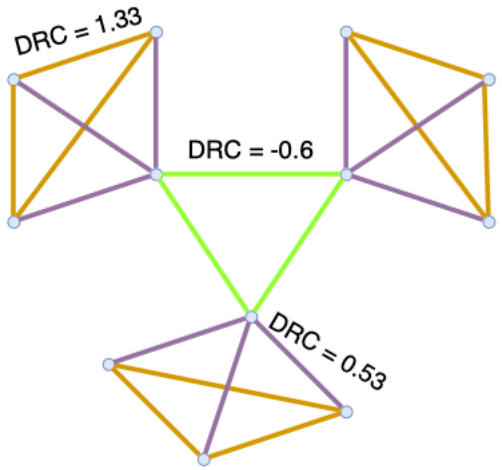


Figure 1: Illustration of DRC on a small graph. Edges in the same color have the same DRC value because of symmetry. Dense connections(yellow edges) are corresponding to positive DRC, while sparse connections(green edges) have negative DRC.

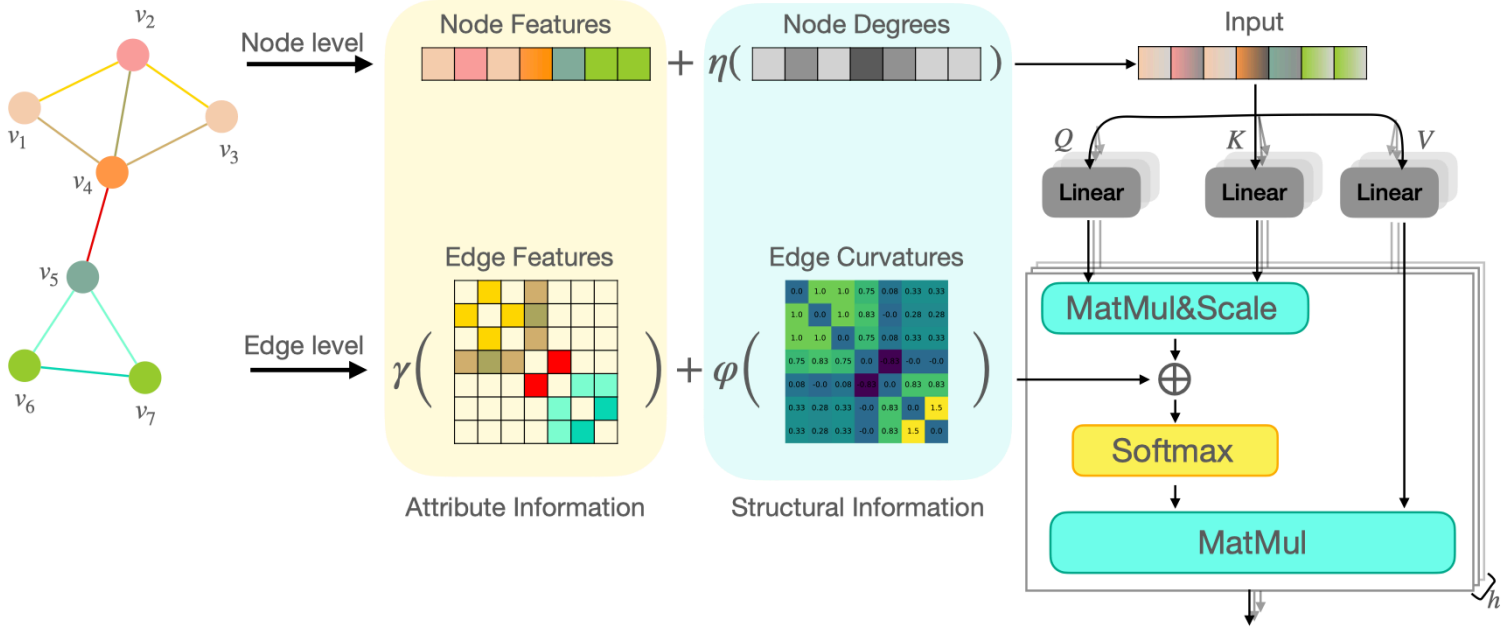


Figure 2: Illustration of Curvphormer with attribute/structure encodings. The input is a combination of two types of node level information, i.e., node features and node degree encoding. Edge level information, i.e., encodings of edge features and curvatures, describes interactions between node pairs, therefore these two encodings are added to the multi-head self-attention module as a bias of the attention weights.

- Discrete Ricci curvature(DRC)

[1] Anonymous. "**Does Structural Information Have Been Fully Exploited in Graph Data?**" In *ICLR 2023 Submission*, 2022

Relational Transformer (RT), ICLR 2023 Submission

CLRS Algorithmic Reasoning Benchmark

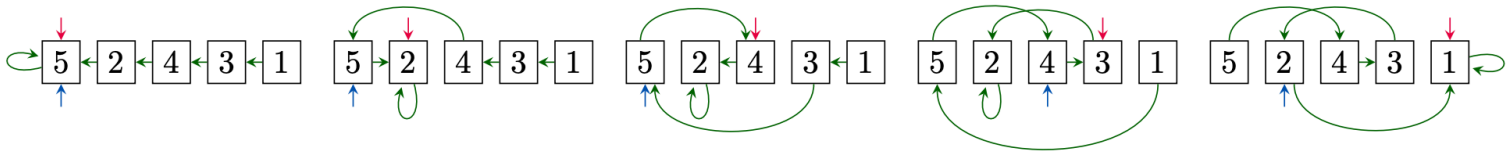
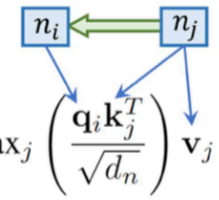


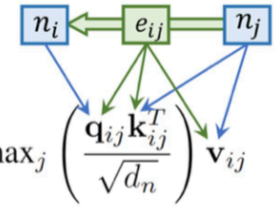
Figure 2. A sequence of hints for insertion sorting a list [5, 2, 4, 3, 1]. **Green** pointers correspond to the predecessor pointers (specifying the list's state throughout the algorithm's execution. Note how the head of the list always points to itself, by convention. Further, note how, at every step, the list is rewired such that the node selected by the **blue** pointer (slot) will point to the current iterator (pointed in **red**).

- Transformer that conditions node updates on edges, also updates edge features, all in O(n^2) time (vs. naive O(n^3) version). Works well on algorithmic reasoning tasks.

Relational Attention



$$\mathbf{a}_i = \text{softmax}_j\left(\frac{\mathbf{q}_i \mathbf{k}_j^T}{\sqrt{d_n}}\right)\mathbf{v}_j$$



$$\mathbf{a}_i = \text{softmax}_j\left(\frac{\mathbf{q}_{ij} \mathbf{k}_{ij}^T}{\sqrt{d_n}}\right)\mathbf{v}_{ij}$$

$$\mathbf{q}_{ij} = [\mathbf{n}_i, \mathbf{e}_{ij}]\mathbf{W}^Q \qquad \mathbf{k}_{ij} = [\mathbf{n}_j, \mathbf{e}_{ij}]\mathbf{W}^K \qquad \mathbf{v}_{ij} = [\mathbf{n}_j, \mathbf{e}_{ij}]\mathbf{W}^V \tag{4}$$

Efficient Implementation

$$\mathbf{q}_{ij} = \left(\mathbf{n}_i \mathbf{W}_n^Q + \mathbf{e}_{ij}\mathbf{W}_e^Q\right) \qquad \mathbf{k}_{ij} = \left(\mathbf{n}_i \mathbf{W}_n^K + \mathbf{e}_{ij}\mathbf{W}_e^K\right) \qquad \mathbf{v}_{ij} = \left(\mathbf{n}_i \mathbf{W}_n^V + \mathbf{e}_{ij}\mathbf{W}_e^V\right) \tag{5}$$

Edge Update

General Version

$$\mathbf{e}_{ij}^{l+1} = \phi_e\left(\mathbf{e}_{ij}^l, \mathbf{e}_{ji}^l, \mathbf{n}_i^{l+1}, \mathbf{n}_j^{l+1}\right) \tag{6}$$

Implementation

$$\mathbf{m}_{ij}^l = \text{ReLU}\left(\text{concat}\left(\mathbf{e}_{ij}^l, \mathbf{e}_{ji}^l, \mathbf{n}_i^{l+1}, \mathbf{n}_j^{l+1}\right)\mathbf{W}_4\right) \tag{7}$$

$$\mathbf{u}_{ij}^l = \text{LayerNorm}\left(\mathbf{m}_{ij}^l \mathbf{W}_5 + \mathbf{e}_{ij}^l\right) \qquad \mathbf{e}_{ij}^{l+1} = \text{LayerNorm}\left(\text{ReLU}(\mathbf{u}_{ij}^l \mathbf{W}_6)\mathbf{W}_7 + \mathbf{u}_{ij}^l\right) \tag{8}$$

[1] Petar Veličković, Adrià Puigdomènech Badia, David Budden, Razvan Pascanu, Andrea Banino, Misha Dashevskiy, Raia Hadsell, Charles Blundell. "**The CLRS Algorithmic Reasoning Benchmark**." In *ICML*. 2022

[2] Anonymous. "**Relational Attention: Generalizing Transformers for Graph-Structured Tasks**." In *ICLR 2023 Submission*, 2022

Specformer, ICLR 2023 Submission

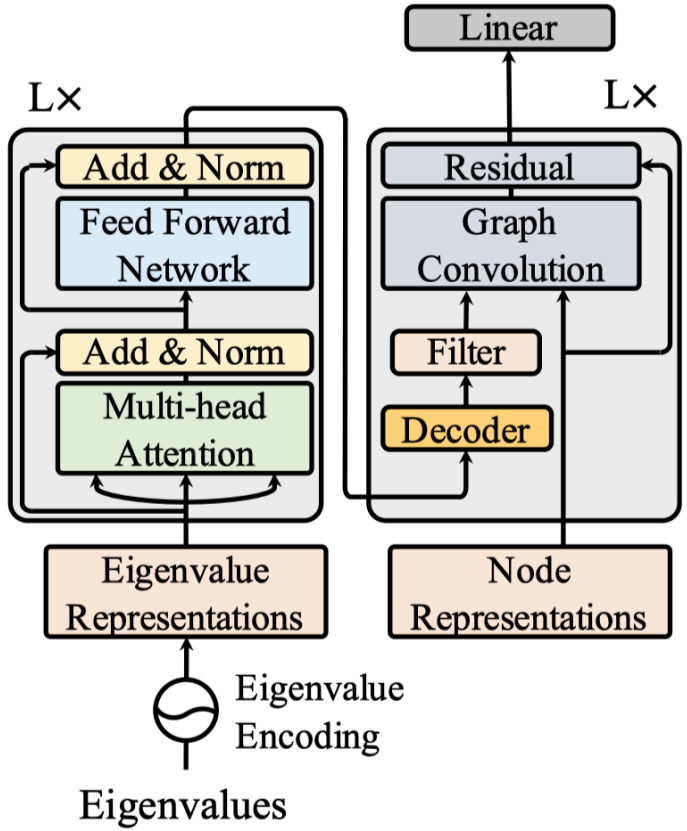


Figure 1: Illustration of the proposed Specformer.

Eigenvalue Encoding

$$\boldsymbol{Z} = [\lambda_1 \| \rho(\lambda_1), \cdots, \lambda_n \| \rho(\lambda_n)]^\top \in \mathbb{R}^{n \times (d+1)}.$$

$$\tilde{\boldsymbol{Z}} = \mathrm{MHA}(\mathrm{LN}(\boldsymbol{Z})) + \boldsymbol{Z},$$

$$\hat{\boldsymbol{Z}} = \mathrm{FFN}(\mathrm{LN}(\tilde{\boldsymbol{Z}})) + \tilde{\boldsymbol{Z}}.$$

Eigenvalue Decoding

$$\boldsymbol{Z}_m = \mathrm{Attention}(\boldsymbol{Q}\boldsymbol{W}_m^Q, \boldsymbol{K}\boldsymbol{W}_m^K, \boldsymbol{V}\boldsymbol{W}_m^V),\ \boldsymbol{\lambda}_m = \phi(\boldsymbol{Z}_m \boldsymbol{W}_\lambda),$$

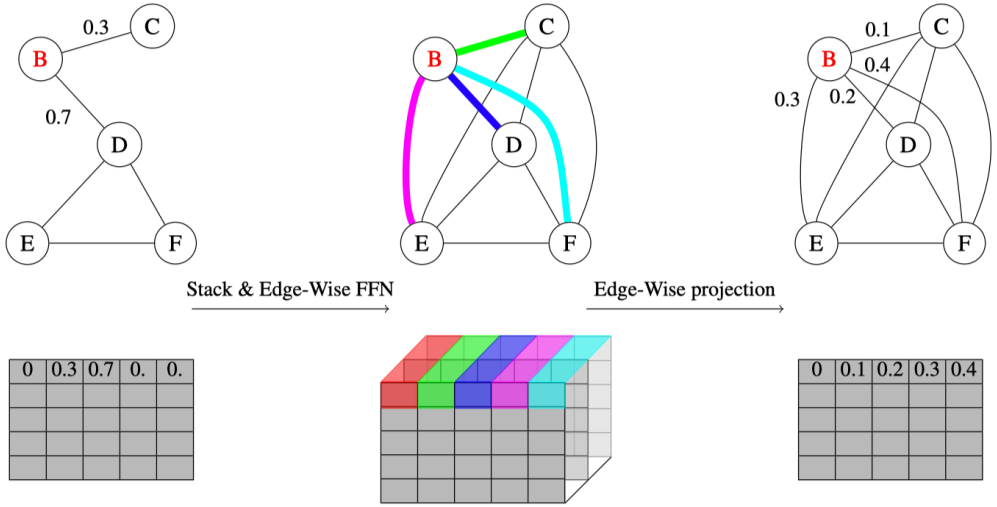
$$\boldsymbol{S}_m = \boldsymbol{U}\mathrm{diag}(\boldsymbol{\lambda}_m)\boldsymbol{U}^\top,\ \hat{\boldsymbol{S}} = \mathrm{MLP}([\boldsymbol{I}_n || \boldsymbol{S}_1 || \cdots || \boldsymbol{S}_M]),$$

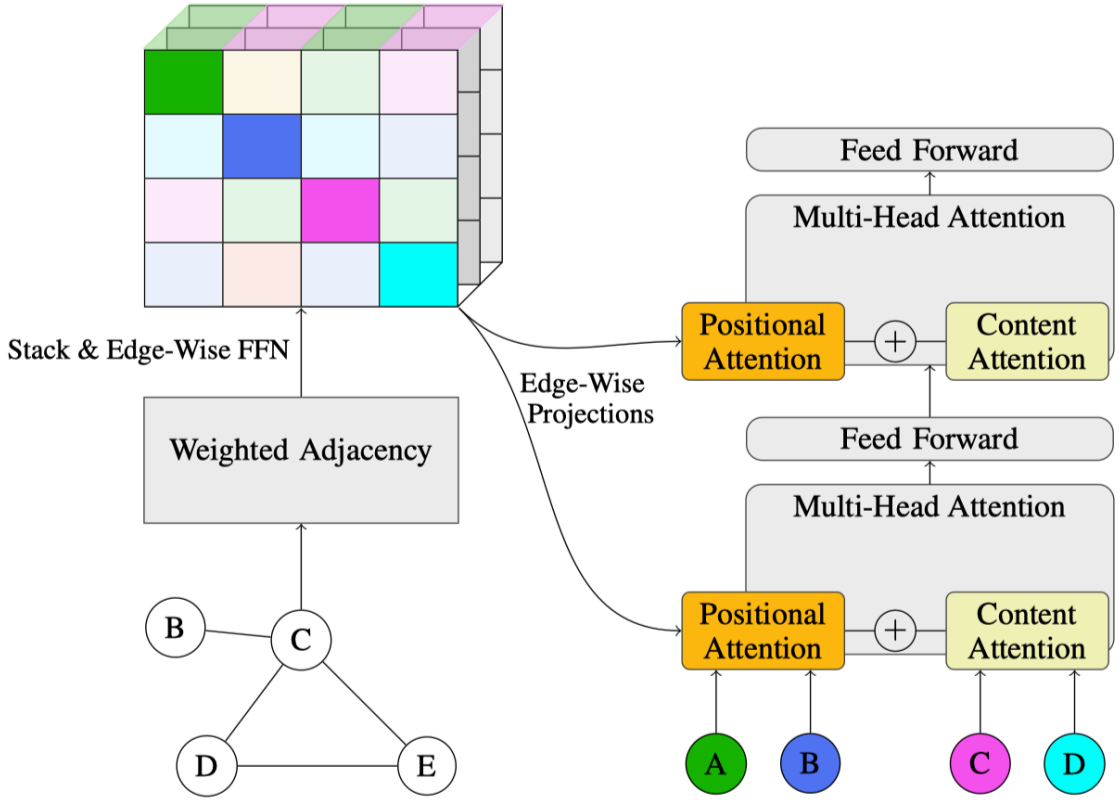
Graph Convolution

$$\hat{\boldsymbol{X}}_{:,i}^{(l-1)} = \hat{\boldsymbol{S}}_i \boldsymbol{X}_{:,i}^{(l-1)},\ \boldsymbol{X}^{(l)} = \sigma\left(\hat{\boldsymbol{X}}^{(l-1)} \boldsymbol{W}_x\right) + \boldsymbol{X}^{(l-1)},$$

[1] Anonymous. "**Specformer: Spectral Graph Neural Networks Meet Transformers**." In *ICLR 2023 Submission*, 2022

~~GD, ICLR 2023 Submission~~





•Modifies node-wise attention matrix with learned relative PEs from powers of a learned weighted adjacency.

[1] Anonymous. "**Diffusing Graph Attention**." In *ICLR 2023 Submission*, 2022

AutoGT, ICLR 2023 Submission

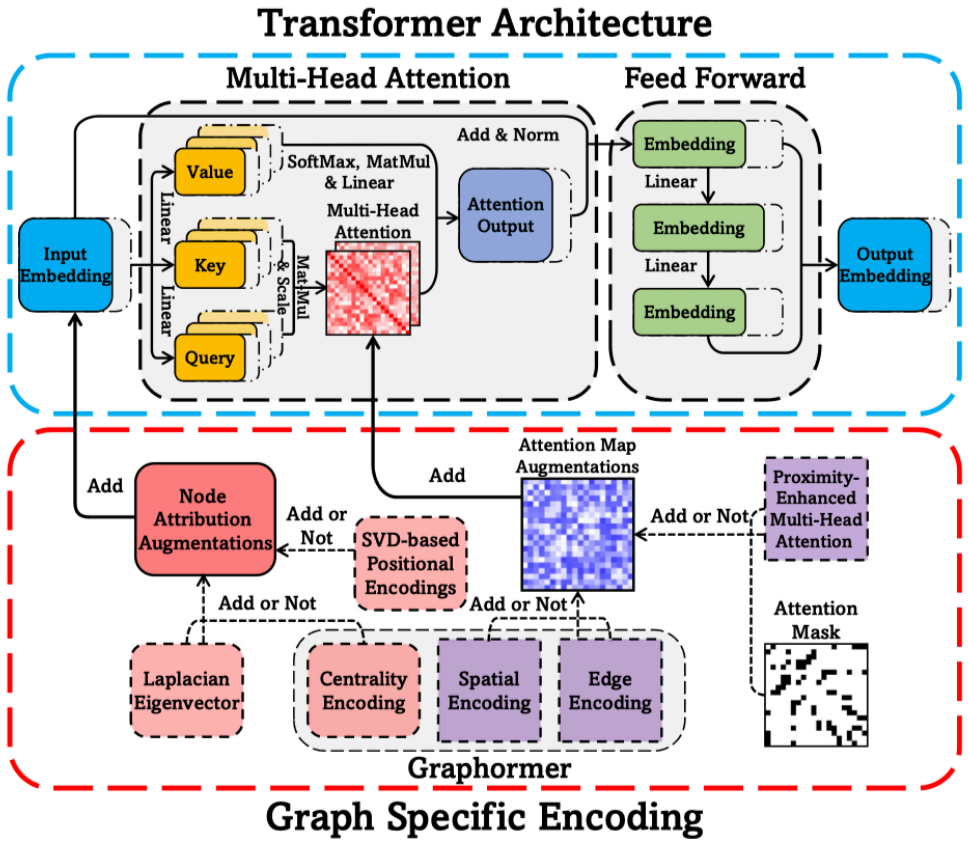
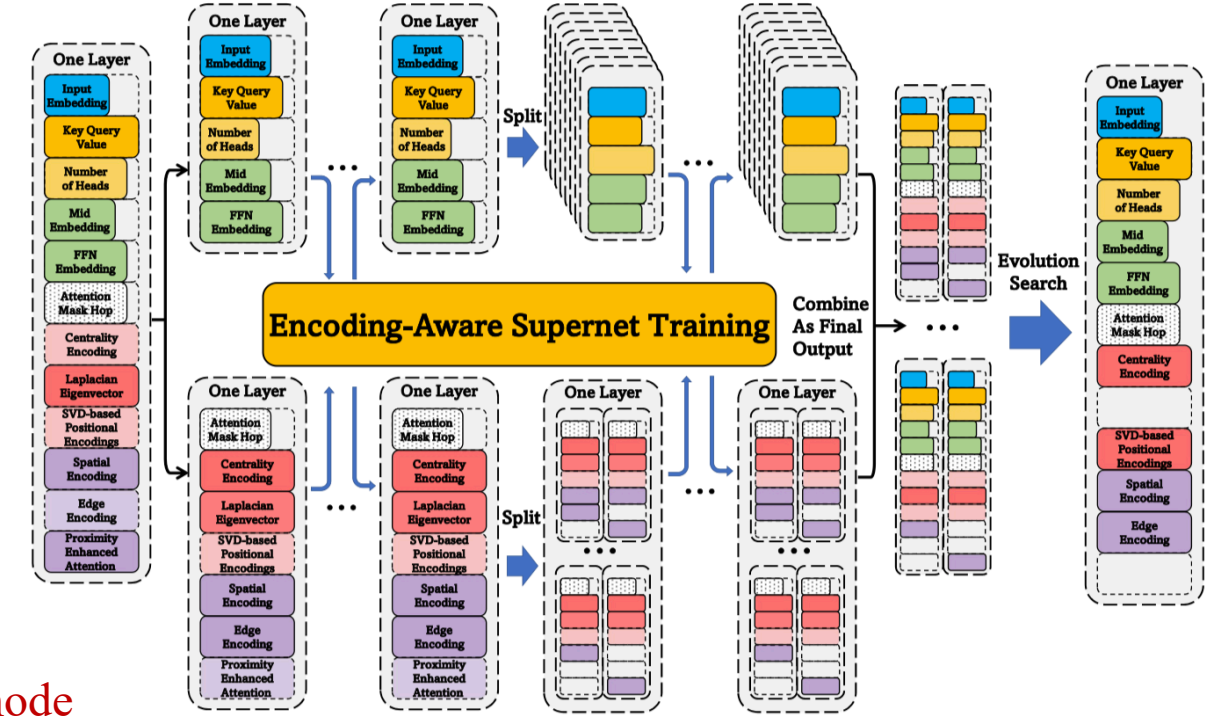


Figure 1: The unified graph Transformer search space

Table 1: The Transformer Architecture Search Space for $\text{AutoGT}_{\text{base}}$ and AutoGT.

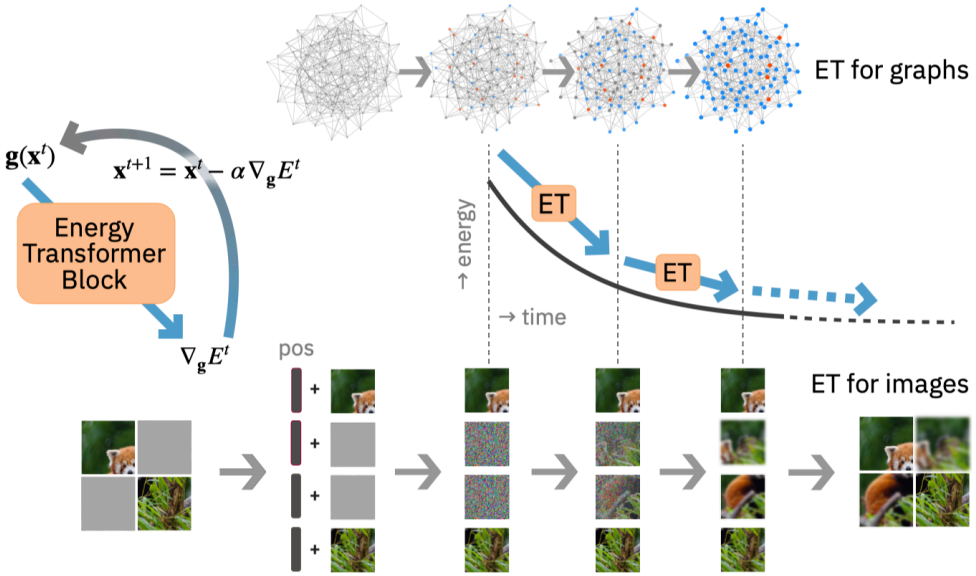
	$\text{AutoGT}_{\text{base}}$		AutoGT	
	Choices	Supernet Size	Choices	Supernet Size
#Layers	{2,3,4}	4	{5,6,7,8}	8
Input Dimension d	{24,28,32}	32	{96,112,128}	128
Intermediate Dimension d_t	{24,28,32}	32	{96,112,128}	128
Hidden Dimension d_h	{24,28,32}	32	{96,112,128}	128
#Attention Heads	{2,3,4}	4	{6,7,8}	8
Attention Head Dimension d_k	{6,8}	8	{12,14,16}	16

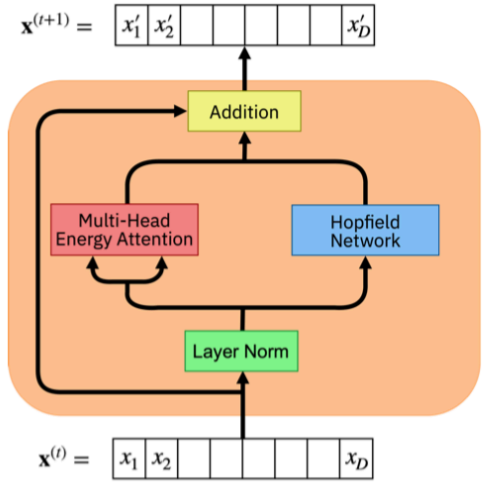


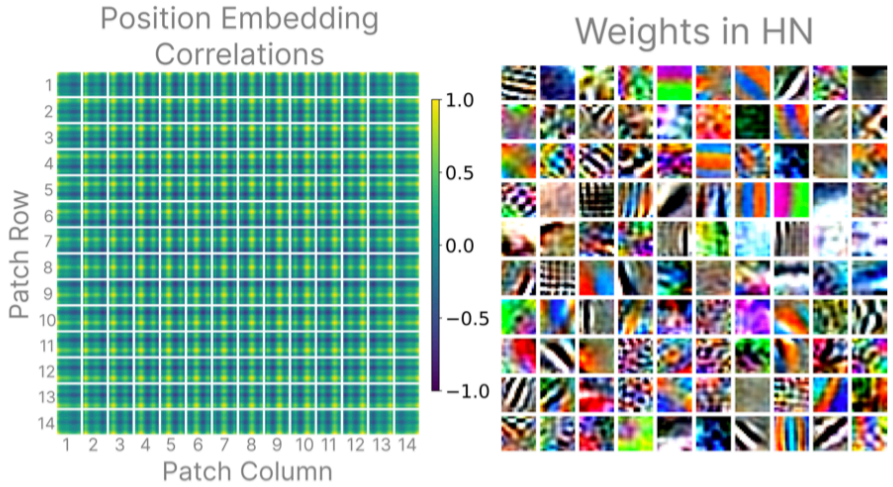
- Neural architecture search for Graph Transformers across node PE, relative PE, and layers / dimensions / heads.

[1] Anonymous. "**AutoGT: Automated Graph Transformer Architecture Search**." In *ICLR 2023 Submission*, 2022

~~Energy Transformer, ICLR 2023 Submission~~







[1] Anonymous. "**Energy Transformer**." In *ICLR 2023 Submission*, 2022

Future Directions

- Model Architecture Design (Tokenization, PE)
  - General
  - Specific: Tree, Hyperbolic, ~~Heterogenous, Relational, Heterophilic~~, Hyper
- Scalability (Sampling Strategy)
- Expressive Power
- Reasoning
- Generalization
- Spectral
- Self-Supervised, Pre-Training
- ~~Substructure~~
- ~~Structure Learning~~
- Data Augmentation
- Applications
  - 4Science